AD-A009 148

TAKEOFF AND LANDING ANALYSIS COMPUTER PROGRAM (TOLA). PART IV. PROGRAMMER'S MANUAL

Fay O. Young, et al

Air Force Flight Dynamics Laboratory
Wright-Patterson Air Force Base, Ohio

January 1975

ACCESSION for	
NTIS	White Section ☑
DDC	Buff Section ☐
UNANNOUNCED	
JUSTIFICATION	
BY..................	
DISTRIBUTION	
Dist.	A
P	

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed and cleared for open publication and/or public release by the appropriate Office of Information (OI) in accordance with AFR 190-17 and DOD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER:

for Alfred G. Draper

PHILIP P. ANTONATOS
Chief, Flight Mechanics Division
Air Force Flight Dynamics Laboratory

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/23 April 1975 - 200

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFFDL-TR-71-155, Part IV	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: AD-A009 148	
4. TITLE (and Subtitle): Takeoff and Landing Analysis Computer Program(Tola) Pt.IV - Programmer's Manual	5. TYPE OF REPORT & PERIOD COVERED: Final report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Fay O. Young, John J. Dueweke	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Proj 1431, Task 143109
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio 45433	12. REPORT DATE: January 1975
	13. NUMBER OF PAGES: 199
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

D D C
MAY 8 1975
D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

PRICES SUBJECT TO CHANGE

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

1. Takeoff and Landing Analysis
2. Computer Program
3. Glide Slope
4. Flare
5. Landing Roll
6. Takeoff Roll
7. Landing Gear Loads and Dynamics
8. Vehicle Control

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A well-defined integration of the various aspects of the aircraft takeoff and landing problem is presented in the form of a generalized computer program. Total aircraft system performance is evaluated during the glide slope, flare, landing roll, and takeoff. The flight dynamics of a generalized, rigid-body, aerospace vehicle are formulated in six degrees of freedom. A flat, nonrotating Earth is assumed. The independent equations of motion of up to five oleo-type landing gears are also formulated. A control management formulation is developed to automatically adjust control variables to correct errors in the vehicle's dynamic state. Stability in the small is used to maintain stability in the large. The equations of motion are integrated using a generalized variable-step Runge-Kutta technique. The formulation is programmed for the CDC CYBER 70 and 6000 series computers using the SCOPE 3.4 operating system. The entire program is written in Fortran Extended.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AFFDL-TR-71-155
PART IV

FOREWORD

Work described in this report was accomplished by the Flight Mechanics Division of the Air Force Flight Dynamics Laboratory and the Digital programming section, 4950th Test Wing under Project 1431, "Flight Path Analysis," Task 143109, "Trajectory and Motion Analysis of Flight Vehicles." The formulation and interim documentation were completed by Major Urban H. D. Lynch. Programming was accomplished by Mr. Fay O. Young of the Digital Programming Section (ADDP), Computer Science Center 4950th Test Wing.

This report was prepared by Mr. John J. Dueweke of the High Speed Aero Performance Branch (FXG), and Mr. Fay O. Young, and combines the applicable portions of FDL-TDR-64-1, Part I, Volume 1, with the interim documentation. The overall report is divided into four parts:

Part I. Capabilities of the Takeoff and Landing Analysis Computer Program

Part II. Problem Formulation

Part III. User's Manual

Part IV. Programmer's Manual

This report was submitted by the authors in March 1972.

AFFDL-TR-71-155
PART IV

ABSTRACT

A well-defined integration of the various aspects of the aircraft takeoff and landing problem is presented in the form of a generalized computer program. Total aircraft system performance is evaluated during the glide slope, flare, landing roll and takeoff.

The flight dynamics of a generalized, rigid-body, aerospace vehicle are formulated in six degrees of freedom. A flat, nonrotating Earth is assumed. The independent equations of motion of up to five oleo-type landing gears are also formulated.

A control management formulation is developed to automatically adjust control variables to correct errors in the vehicle's dynamic state. Stability in the small is used to maintain stability in the large.

The equations of motion are integrated using a generalized variable-step Runge-Kutta technique.

The formulation is programmed for the CDC CYBER 70 and 6000 series computers using the SCOPE 3.4 operating system. The entire program is written in Fortran Extended.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTD)

SECTION I

INTRODUCTION

In the design of an aircraft, the engineer is confronted with the problem of takeoff and landing and the design of aircraft systems and techniques to perform this function. The final evaluation of these systems lies in the answer to the question: How does the aircraft and its systems perform as a unit? The Takeoff and Landing Analysis (TOLA) Computer Program is the result of an attempt to generalize the aircraft, the main aircraft control systems, and the landing-takeoff situation into a single comprehensive calculation to answer this question.

The TOLA simulation answers the above question in the form of a well-defined integration of the various aspects of takeoff and landing. In the equations of motion the assumption is made that the main aircraft frame is rigid; however, the dynamic effects of up to five independent landing gears are included in the equations. The position and velocity of each strut and secondary piston are obtained by numerical integration subject to position constraints (for example, the main strut must move within the limits of the fully extended position and strut bottoming position). The same form of solution applies to the aircraft itself.

SECTION II

COMPUTER AND SYSTEM REQUIREMENTS

The Takeoff and Landing Analysis Computer Program (TOLA) has been written for use with the CDC CYBER 70 and 6000 series computers using the SCOPE 3.4 operating system. The entire TOLA is written in Fortran Extended.

1. CYBER 70 or 6000 series computer

   a. A CYBER 70 or 6000 series computer with 32K (decimal), or larger core.

   b. Six CDC tape transports.

   c. Control Data Card Reader.

   d. Control Data Printer.

   e. 12 inch CALCOMP off-line plotter.

2. The CDC tape transports may be replaced by other equipment which will simulate magnetic tapes such as disk storage, except for one tape unit that may be used to generate a plot tape for the CALCOMP off-line plotter.

SECTION III

PROGRAMMING CONCEPTS

1. THE USE OF COMMON

Whenever possible, a variable is placed in the FORTRAN "COMMON" Area. There are several reasons for this:

a. The communications between subroutines is simplified.

b. The structure of the directory is simplified. Since the number of variables in COMMON is quite large, all COMMON cards are not placed in each assembly/compilation. Instead, required "dummy" cards are placed in each deck of source cards. This has in a small manner reduced the number of COMMON cards in each deck.

2. TABLES AND TABLE USAGE

One of the usual required modifications of any program is the change of table sizes. With this in mind, a COMMON block of locations has been set aside and the required number of cells for each table is specified with data (see TABRE for data preparation). This requires no reassembly or recompilation unless the total number of cells required exceeds the COMMON block of 800 cells.

3. SYMBOLIC INPUT

Although the FORTRAN system itself has a system of input routines, the program does the actual translation of the cards using special coded routines. Input data may be read using a system of symbols which is designed to give engineering meaning to the analyst. The symbols are referenced to actual locations by the use of COMMON and subscripts.

4. TRAJECTORY PRINTING METHOD

The printing of a trajectory may be divided into four categories.

a. Initial Printing - The printing of specific values at the first stage and at each subsequent major stage.

b. Code Printing - The printing of codes which will identify the variables which are to be obtained in the coming time history print.

c. Time History Printing - The printing of values specified at the requested points of the trajectory.

d. Diagnostic Error Printing - The printing of errors detected by the program.

All input data involved for a case is printed on the output page preceding the computation of the first stage printout. Also, data read in at stage times will be printed out between the stages of the trajectory output.

Initial print is designed to print certain values which will be constant during the trajectory and serves as a reminder of what values have been used for these constants.

Code printing is performed once per major stage to identify the time history.

The time history print is designed to print in a minimum space. That is, if a certain variable is not desired as output, it is not printed and other desired variables are moved in the print format accordingly.

The entire printing is controlled to print on a page 11 x 14 inches and will print a maximum of 51 lines per page. Page ejection and lines control are provided by the subroutines DEF and LINES.

5. TAPE USAGE

Tapes	Equipment	Usage
Tape 5	Disk or Tape	Data Input
Tape 6	Disk or Tape	Printed Output
Tape 13	Disk or Tape	Data saved to be used to generate a plot tape by Plot program (PLTSDF)
Tape 16	Disk or Tape	Used by the symbolic input routine to save input data
Tape 31	Disk or Tape	
Tape 7	Tape	CALCOMP plot tape

The above describes the tape usage other than for the FORTRAN system. All modification of tapes required may be made with control cards placed in front of the program before submitting to the computer.

6. STRUCTURE OF PROGRAM

Due to core storage limitations (32K), it was necessary to use the Overlay feature. The following is the structure of the program:

a. OVERLAY (0,0) (FOR. EXT)

TOLA	STFL	TRNPOS
EXE	STFLD	HIHO
INUPD	STOVAR	TLU
LNUPD	ARRAY	TFFS1

INPUZ	LINES	
INTEG	ASIN	
UPDAT	ACOS	VPCS1
MIMIN	ATAN2	SACS1
LGDET	ERROR	AERO1
STGTS1	EXERR	AQUAD
STGTST	NDTLU	OPT1
TDATA	ATMS	LGEAR1
	INVR3	LGEA3C
ASRCH	MULT31	SDFLGP
DEF		

b. <u>OVERLAY (1,0) (FOR EXT)</u>

TOLAN1	TSRCH
TFFS2	DSERCH
READ	PACKRR
DIPLAC	RITE
TABRE	DIR0DA
READA	DIR1DA
STORE	DIR2DA
WRCARD	

c. OVERLAY (2, 0) (FOR EXT)

TOLAN2	ENGREV
AUTS	ENGFL
FLARE1	CENGL
AUTPR1	CTENGL
THAUTS	TFFS8
TFFS9	

7. PROGRAM ORGANIZATION

The TOLA Computer Program is written in FORTRAN Extended. The program is segmented and takes advantage of the FORTRAN overlay features.

This section attempts to describe the overall organization of the program from the viewpoints of control cards, tape usage, deck set-up, and organization.

The program is broken up into three overlays as follows:

a. Overlay (0, 0). Contains all system routines, main, executive, integration, computation of the equations of motion, and printing.

b. Overlay (1, 0). Set up tables, input routines, and input directory.

c. Overlay (2, 0). All routines of the Autopilot.

Plotting tapes are generated by a separate program for plotting on the Cal Comp plotter.

a. Storage Reference. All variables requiring arrays have been arranged in the standard FORTRAN convention; for example, an array A_i is stored in increasing storage locations for increasing i. Matrices are stored columnwise.

b. Integers. All integers are assumed to be in a 60-bit word, right-justified.

c. CØMMØN. In order to decrease the length and time required in calling sequences, liberal use of labeled CØMMØN has been made. For the actual variable and their arrangement in CØMMØN, the user is referred to the program listing.

d. Variable Names. Because any variable may be referred to by FORTRAN, all integer variable names begin with the leading letters I, J, K, L, M, or N. This does not mean that all noninteger variable names begin with letters other than I, J, K, L, M, or N. They may, in some subprogram, be declared integer or real.

8. DECK SETUP

a. Running the TOLA Program requires a particular deck setup. The deck structure is presented as a guide only in determining this setup.

b. CONTROL CARDS (CDC CYBER 70 or 6000 Series, SCOPE 3.4). All control cards are left justified in column 1. The end of record is a 7, 8, 9 punched in column 1 and on end of job card is a 6, 7, 8, 9 punch in column 1. In the control card examples an end of record and an end of job will be used in place of the cards.

(1) The following control cards will execute the TOLA Computer Program from an UPDATE tape and not print a listing of TOLA.

Job Card

LABEL, ØLDPL, R, L = TØLACP, VSN = tape No. RING ØUT

UPDATE, F.

FTN, I = CØMPILE, L = 0.

RETURN, ØLDPL.

LDSET, PRESET = ZERO.

LØAD, LGØ.

NØGØ.

TØLA.

End of Record

Changes to TØLACP if any in UPDATE format

End of Record

Data Cards

End of job.

If a listing is desired, omit the parameter L = 0 on the FTN control card.

(2) The following control cards will generate a new UPDATE tape, an absolute file on tape, and list the TØLA Program.

Job Card

LABEL, ØLDPL, R, L = TØLACP, VSN = tape No. RING ØUT

LABEL, NEWPL, W, L = TØLACP, VSN = tape No. RING IN

UPDATE, N, F.

FTN, I = CØMPILE.

RETURN, ØLDPL.

UNLØAD, NEWPL.

LABEL, TØLA, W, L = TØLACPABS, VSN = tape No. RING IN

LDSET, PRESET = ZERØ.

LØAD, LGØ.

NØGØ.

TØLA.

End of record

Changes to TØLA if any in UPDATE format

End of Record

Data cards

End of job

(3) The following control cards will execute from an absolute file on tape. TØLA is an absolute file on a tape.

Job card

LABEL, TØLA, R, L = TØLACPABS, VSN = tape No. RING ØUT

TØLA.

End of record

Data cards

End of job

(4) If it is desired to save data on tape for CALCθMP plotting, include the following control card with the LABEL cards:

LABEL, TAPE 13, W, L = TθLADATA, VSN = tape No. RING IN

(5) The following control cards will generate a new updated program on permanent file (PF), generate on absolute file of TθLA on PF, and execute TθLA.

Job Card

LABEL, θLDPL, R, L = TθLACP, VSN = tape No. RING θUT

REQUEST, NEWPL, *PF.

UPDATE, N, F.

RETURN, θLDPL.

CATALθG, NEWPL, TθLACP, RP = 999, CY = 1, ID = Prob No.

RETURN, NEWPL.

F TN, I = CθMPILE, L = 0

REQUEST, TθLAP, *PF.

LDSET, PRESET = ZERθ.

LθAD, LGθ.

NθGθ.

TθLA.

CATALθG, TθLAP, TθLACP, RP = 999, CY = 2, ID = Prob No.

End of record

Changes to TθLA if any in UPDATE format

End of record

Data cards

End of job

(6) To execute from a permanent file, use the following control cards:

Job Card

ATTACH, TØLA, TØLACP, CY = 2, ID = XXXXXX.

TØLA.

End of record

DATA cards

End of job

(7) To execute from a permanent file, and generate a plot tape for the CALCØMP plotter on TAPE 7, use the following control cards. The plot tape generating program (PLTSDF) is located on PF: PLTSDF, CY = 1.

Job Card

ATTACH, TØLA, TØLACP, CY = 2, ID = XXXXXX.

TØLA.

RETURN, TØLA

REQUEST, TAPE 7, MT, HI, N, VSN = Tape No. Ring IN

ATTACH, PLTSDF, PLTSDF, CY = 1, ID = XXXXXX.

PLTSDF.

End of record

Data cards for TØLA

End of record

Data cards for PLTSDF

End of job

9. DATA FORMAT

Card Format - The program input routine (READ) expects the following format.

Card Columns -	1 - 6	7	8 - 10	11	12 - 66	67 - 72	73 - 80
Field	I	II	III	IV	V	VI	VIII

Card Field I - Contains the symbolic name of the variable which data contained in Field V begins loading. Example: Card Column

1	12
GAM7D	-1.23
SIG7D	90.

Card Field II   Not Used

Card Field III - Contains the words DEC, ØCT, BCD, TRA, INT, or is blank depending on the type of data to be loaded. The word ØCT indicates that the data is to be interpreted as octal numbers. The word BCD specifies that N binary coded decimal words (N punched in column 12) beginning in column 13 are to be loaded. The word TRA denotes to the input routine that all data has been read and to return control to the calling program. The word DEC and blank are equivalent and specify that data loaded is decimal data.

ØCT Example			
Card Column	1	8	12
	NSMAIN	ØCT	17

BCD Example			
Card Column	1	8	12
	REM	BCD	3SDF2-GEAR-MOD

The 3 in Column 12 specifies 3 words where each word is considered to be 6 characters including blanks. The largest number of 6-character words that can be loaded from one card is 9. The analysts should be very careful to see that the BCD information does not get punched into Field VI. This will cause an input error.

DEC Example			
Card Column	1	8	12
	VTAB01	DEC	2,0.,1.67,20000.,1.67

Note that the first character in Column 12 is an integer; the input routine will load only one integer per DEC card, and that has to be the first number punched in Field V.

VTAB01   DEC   2.,0.,1.67,20000.,1.67

If the above card is punched, the two will now be loaded into the machine as a binary floating point number. The other numbers will be loaded the same, with the decimal point assumed right-justified.

If anything other than ØCT, BCD, INT, TRA, or blank appears in Field II then the word DEC is assumed.

INT Example

Card Column	1	8	12
	IP	INT	1
	IP		1,
	IP	INT	1,1,1,1

When the word INT is used, it is assumed that all numbers on the card will be loaded as integers. If only one integer is punched per card the INT may be punched or omitted.

Card Field IV-Not Used
Card Field V

The actual input data to the program is punched in the Field V. DEC, INT and ØCT numbers must always be left-adjusted; that is, it must always start in column 12 on the input card. All numbers are separated by a "comma" and the field terminates with the first blank. BCD information begins in Column 13 and the maximum number of 6-character words per card is nine. Note that since Field V ends with the first blank, the user may punch any comments in the remainder of the field.

Card Field VI

This field specifies the initial subscript of the data in Field V. If this field is blank, an initial subscript of 1 is implied. The subscript may appear anywhere within the field.

Example Card Column	1	12	67
	PZERØ	30470.4,41538.24,41538.24	1 (or blank)
	PZERØ	42538.24,42538.24	4

In the example above, the number 30470.4 is loaded into the first cell of the array PZERØ. On the second card, 42538.24 is loaded into the fourth cell of the array. The one and four punched in Field VI indicate the subscript for the array PZERØ.

Card Field VII

Not used as far as the input routine is concerned. This may be used as a sequence number for the card.

10. TABLE FORMAT

The various types of tables used by the program may be classed as follows:

<u>One Dimensional Tables</u>

Example 1:	NTIRES	$n_i = f(i), i = 1, 2, \ldots, NSTRUT$
Card Column	1	12
	NSTRUT	5
	NTIRES	4.,6.,6.,6.,6.

INSTRUT= Fixed point number which is the number of struts on the aircraft. For example, the number of tires on strut 2 is 6; i.e., $n_2 = f(2) = 6$.

i = independent variable values

n_i = Corresponding dependent variable values

Example 2:		Aerodynamic Data
Card Column	1	12
	INDA01	1
	ATAB01	.0065,.00748

INDA01 ≠1 designates that there are data in ATAB01. The first data point is for full ground effect; the second data point is for no ground effect in all aerodynamic tables.

Two Dimensional Table

Example:	VTAB01	$x_{cg} = f(m)$
Card Column	1	12
	VTAB01	$N, M_1, X_{CG_1}, M_2, X_{CG_2}, \ldots, M_N, X_{CG_N}$

N = Fixed point number equal to 2 times the number of independent variables. For a 20-point table, N would equal 40. The total number of machine cells required for this table is 41.

M_i = Independent variable values

X_{CG_i} = Corresponding dependent variable values

N - Dimensional Table

Example:	T	$F(N, M_N)$
Card Column	1	12
	IT10W	NN
	IT10X	NMN
	TTAB10	$N_1, N_2, N_3, \ldots, N_{NN}$
	TTAB10	$M_{N1}, M_{N2}, M_{N3}, \ldots, M_{NNMN}$
	TTAB10	$T_{N1}, M_{N1}, T_{N2}, M_{N1}, \ldots, T_{NNN}, M_{N1}$
	TTAB10	$T_{N1}, M_{N2}, T_{N2}, M_{N2}, \ldots, T_{NNN}, M_{N2}$
	.	. . .
	.	. . .
	.	. . .
	TTAB10	$T_{N1}, M_{NNMN}, T_{N2}, M_{NNMN}, \ldots, T_{NNN}, M_{NNMN}$

NN and NMN are fixed point numbers of independent variables. T_{N_1}, $M_{N1}, \ldots, T_{N_{NN}}$, M_{NNMN} are values of independent variables. The table subscripts would apply to the N-dimensional table as well as the two dimensional. The total number of machine cells required for an N-dimensional table equal NN * NMN + NN + NMN.

Examples:

C = F(X, Y)	NX = 2 = points for X
	NY = 2 = points for Y
Machine cells required	2 x 2 + 2 + 2 = 8 cells
C = F(X, Y, Z)	NX = 20 = points for X
	NY = 10 = points for Y
	NZ = 15 = points for Z
Machine cells required	20 X 10 X 15 + 20 + 10 + 15 = 3045 cells

SECTION IV

FORTRAN EXTENDED OVERLAY (0,0)

1. TOLA - MAIN PROGRAM

a. Purpose - Initializes parameters in the read routine through COMMON, initializes table data, and parameters in storage routines, and calls EXE.

b. Usage - Calls the executive routine (EXE) for each case.

2. EXE - EXECUTIVE ROUTINE

a. Purpose - To zero all variables that may be read in as input, initialize subprograms, and set nominal values. Read input, do all the proper initialization, set up tables dimensions, check for staging, printing, etc.

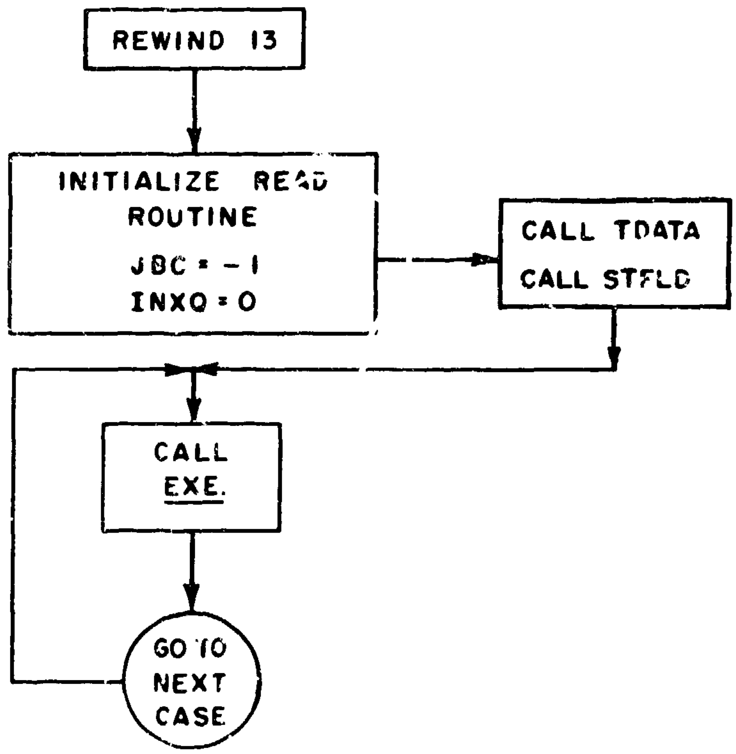
b. Usage - The executive routine is the controlling program. When a case is completed, a return is made to TOLA.

3. STGTSI, STGTST - STAGE TESTING ROUTINES

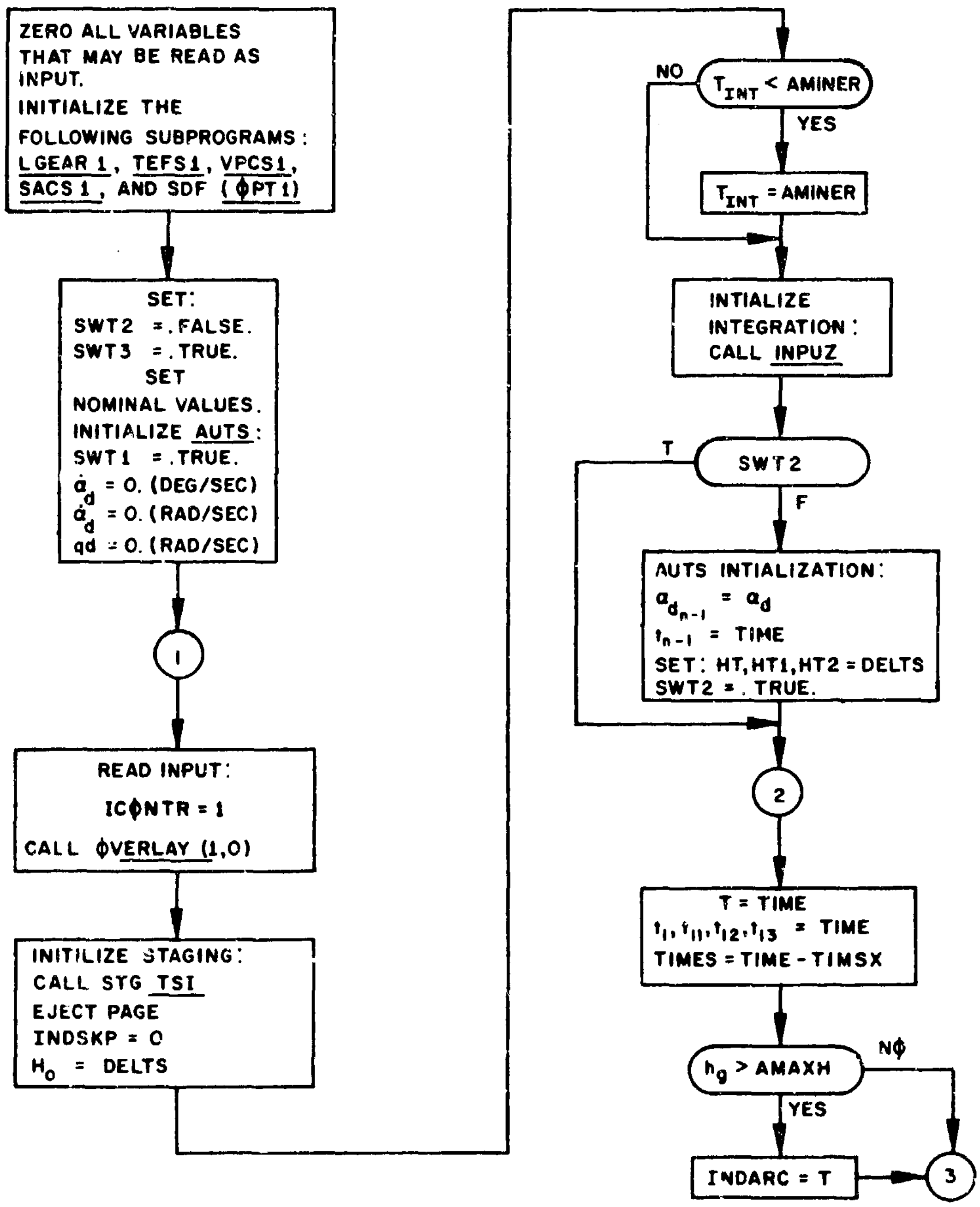
a. Purpose - To test the possiblity of staging on any of up to four increasing and four decreasing variables.

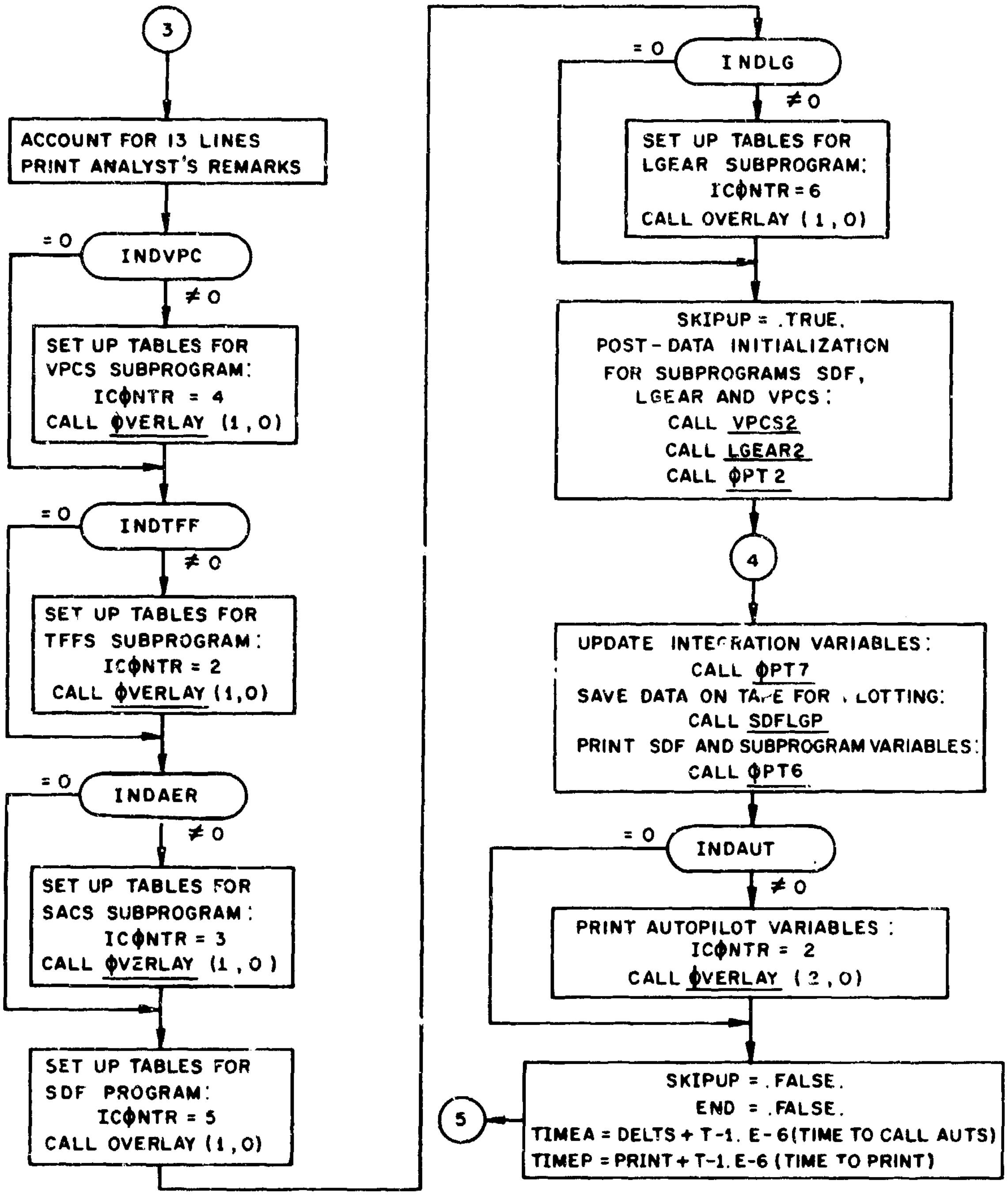
b. Method - Given the four increasing variable BCD names (in array AINCRS) and the four decreasing variable BCD names (in array DECRES) the routine first searches the directory for their location (routine STGTSI).

FLOW DIAGRAM — MAIN PROGRAM (TOLA)

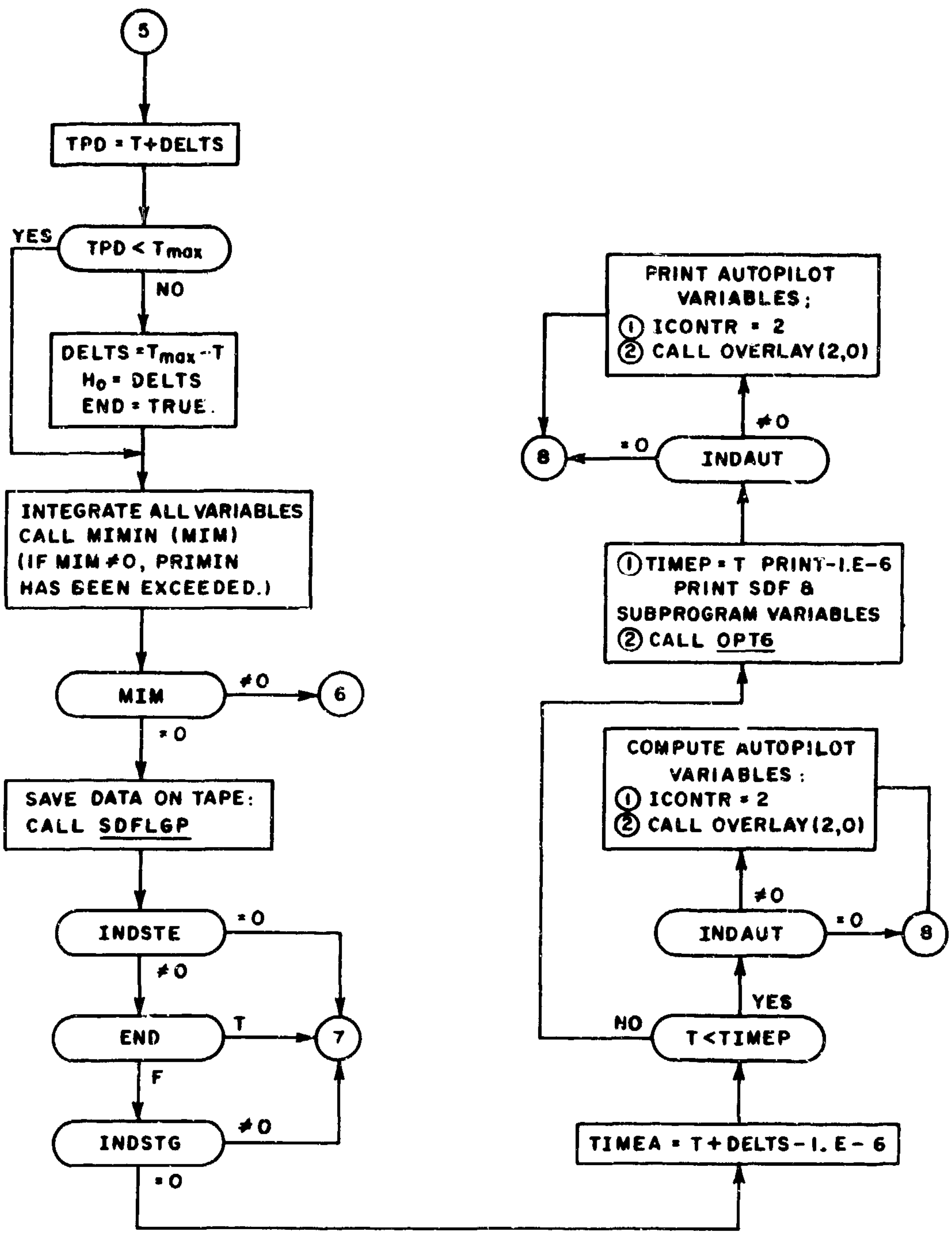


FLOW DIAGRAM - EXECUTIVE ROUTINE (EXE)

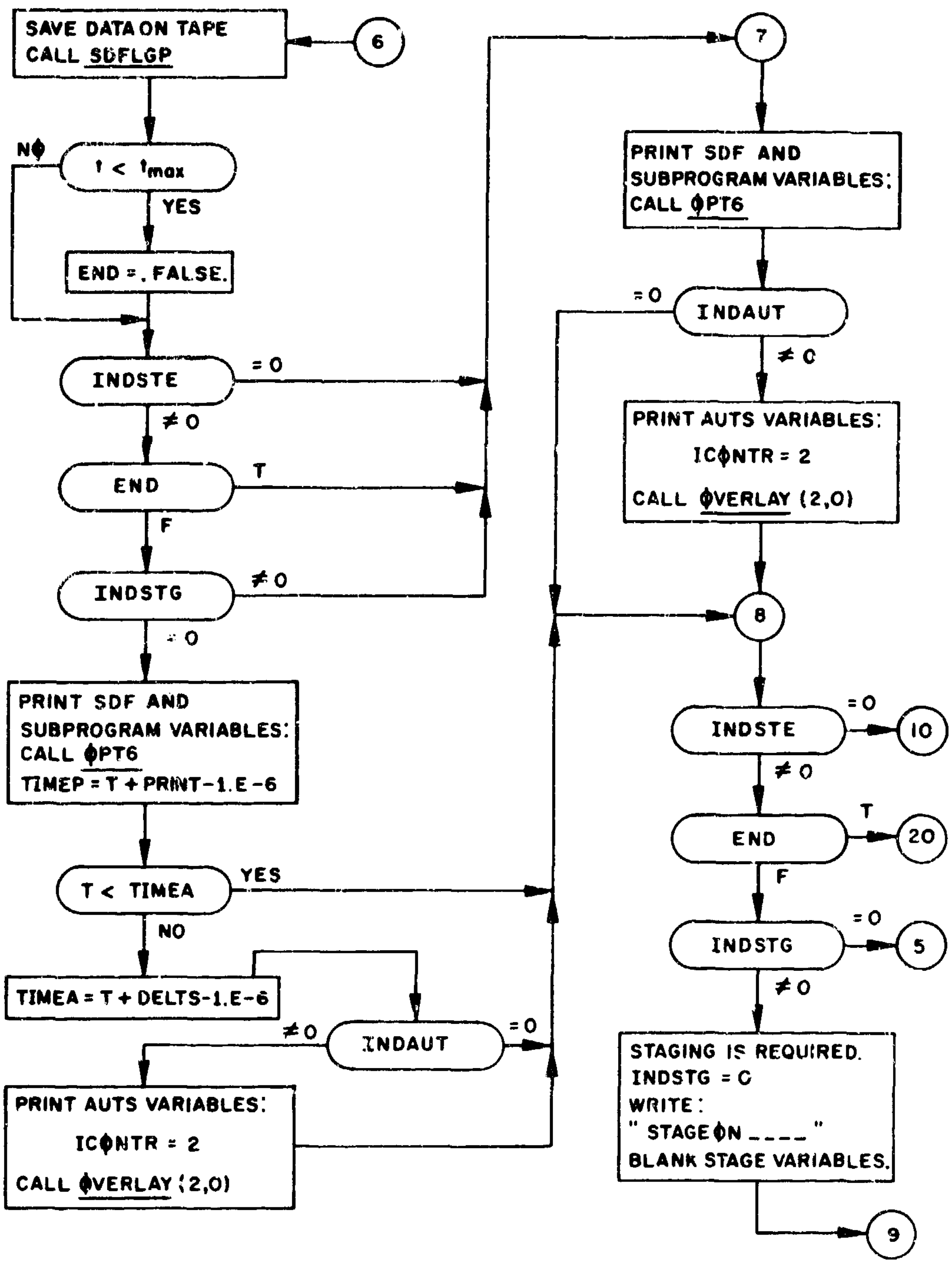


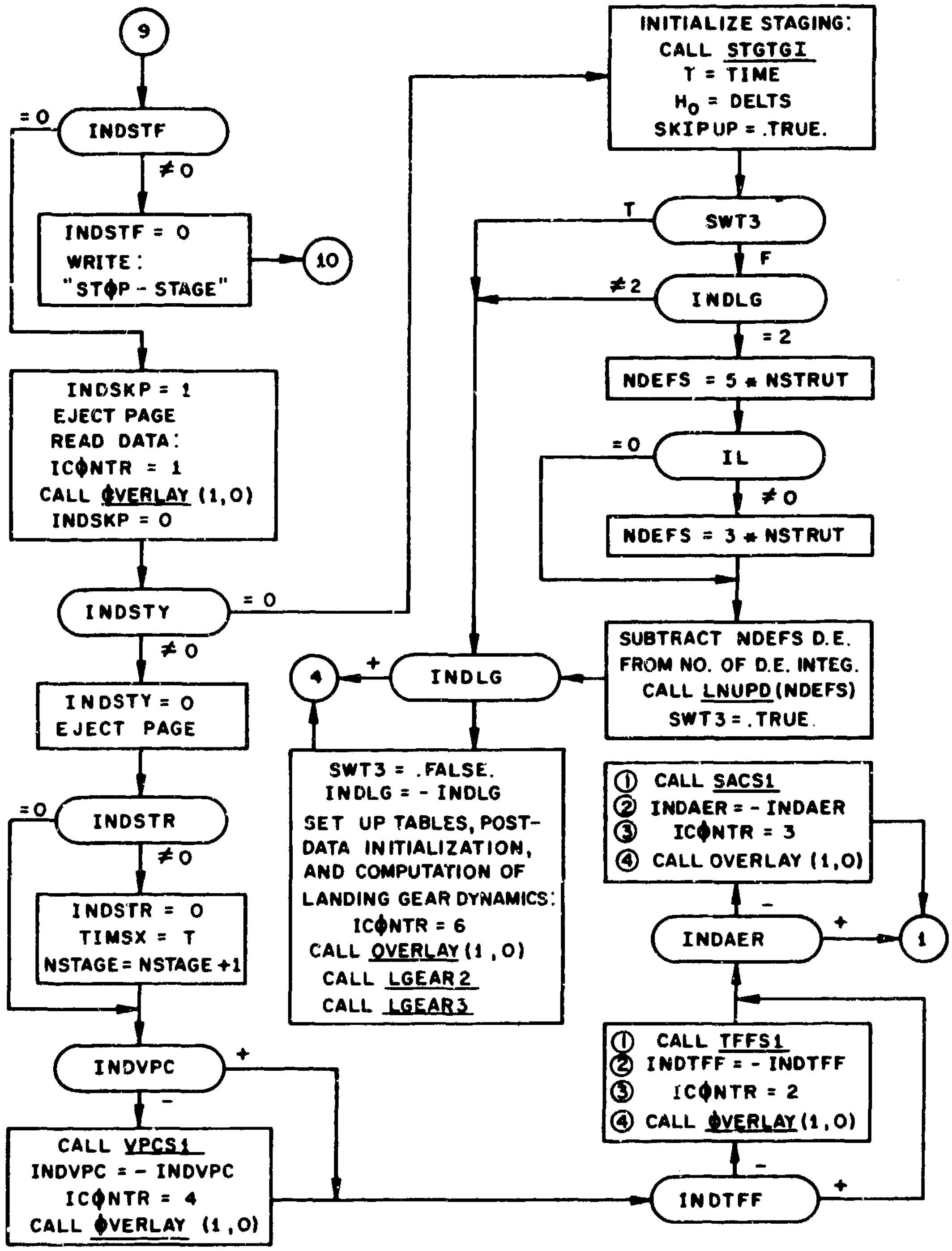

3
ACCOUNT FOR 13 LINES
PRINT ANALYST'S REMARKS
= 0
INDVPC
≠ 0
SET UP TABLES FOR
VPCS SUBPROGRAM:
ICØNTR = 4
CALL ØVERLAY (1, O)
= 0
INDTFF
≠ 0
SET UP TABLES FOR
TFFS SUBPROGRAM:
ICØNTR = 2
CALL ØVERLAY (1,O)
= 0
INDAER
≠ 0
SET UP TABLES FOR
SACS SUBPROGRAM:
ICØNTR = 3
CALL ØVERLAY (1, O)
SET UP TABLES FOR
SDF PROGRAM:
ICØNTR = 5
CALL OVERLAY (1, O)
= 0
INDLG
≠ 0
SET UP TABLES FOR
LGEAR SUBPROGRAM:
ICØNTR = 6
CALL OVERLAY (1, O)
SKIPUP = .TRUE.
POST-DATA INITIALIZATION
FOR SUBPROGRAMS SDF,
LGEAR AND VPCS:
CALL VPCS2
CALL LGEAR2
CALL ØPT2
4
UPDATE INTEGRATION VARIABLES:
CALL ØPT7
SAVE DATA ON TAPE FOR PLOTTING:
CALL SDFLGP
PRINT SDF AND SUBPROGRAM VARIABLES:
CALL ØPT6
= 0
INDAUT
≠ 0
PRINT AUTOPILOT VARIABLES:
ICØNTR = 2
CALL ØVERLAY (2, O)
SKIPUP = .FALSE.
END = .FALSE.
TIMEA = DELTS + T-1. E-6 (TIME TO CALL AUTS)
TIMEP = PRINT + T-1. E-6 (TIME TO PRINT)
5

FLOW DIAGRAM - EXECUTIVE ROUTINE (EXE)

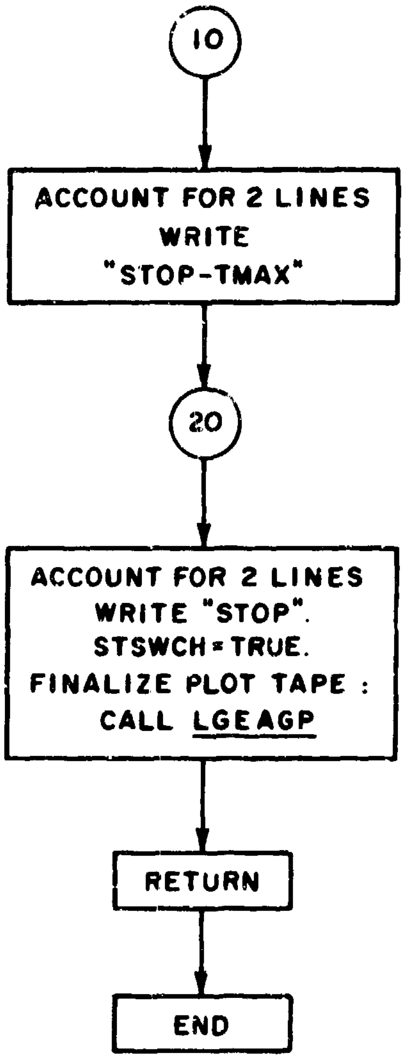


FLOW DIAGRAM-EXECUTIVE ROUTINE (EXE)




9
INDSTF
=0
≠0
INDSTF = 0
WRITE:
"STØP - STAGE"
10
INDSKP = 1
EJECT PAGE
READ DATA:
ICØNTR = 1
CALL ØVERLAY (1,0)
INDSKP = 0
INDSTY
= 0
≠ 0
INDSTY = 0
EJECT PAGE
INDSTR
=0
≠0
INDSTR = 0
TIMSX = T
NSTAGE = NSTAGE +1
INDVPC
+
-
CALL VPCS1
INDVPC = - INDVPC
ICØNTR = 4
CALL ØVERLAY (1,0)
INITIALIZE STAGING:
CALL STGTGI
T = TIME
H0 = DELTS
SKIPUP = .TRUE.
SWT3
T
F
INDLG
≠2
=2
NDEFS = 5 * NSTRUT
IL
=0
≠0
NDEFS = 3 * NSTRUT
SUBTRACT NDEFS D.E.
FROM NO. OF D.E. INTEG.
CALL LNUPD(NDEFS)
SWT3 = .TRUE.
INDLG
+
4
SWT3 = .FALSE.
INDLG = - INDLG
SET UP TABLES, POST-
DATA INITIALIZATION,
AND COMPUTATION OF
LANDING GEAR DYNAMICS:
ICØNTR = 6
CALL OVERLAY (1,0)
CALL LGEAR2
CALL LGEAR3
① CALL SACS1
② INDAER = - INDAER
③ ICØNTR = 3
④ CALL OVERLAY (1,0)
INDAER
-
+
1
① CALL TFFS1
② INDTFF = - INDTFF
③ ICØNTR = 2
④ CALL ØVERLAY (1,0)
INDTFF
-
+

FLOW DIAGRAM – EXECUTIVE ROUTINE (EXE.)



In routine STGTST, and when ISTAGE=0, the routine may then be used to test for any of the increasing variables being greater than the values in the STEST array (i.e., increasing variable$_i$ $\geq$ STESTi). In like manner, the decreasing variable$_i$ $\geq$ STESTDi test is made for i = 1, 2, 3, 4.

Note that the testing stops at the first test to be satisfied. If a test is satisfied for increasing variables, the routine places the BCD name of the varible in STGVAR and the BCD word "INCR" in STGVAR-1. This process is similar for decreasing variables, except that "DECR" is placed in STGVAR-1.

Similarly, when ISTAGE $\neq$ 0, the routine may be used to test for <u>all</u> of the increasing variables being equal to the values in the STEST array (i.e., increasing variable$_i$ = STESTi). In like manner, the decreasing variable$_i$ = STESTDi test is made for i = 1, 2, 3, 4. Note that in order to pass the test the conditions must be met for all i.

c. Usage

(1) Initialization - The statement, CALL STGTSI

must be given. This statement must be given each time a new set of variable names is to be tested.

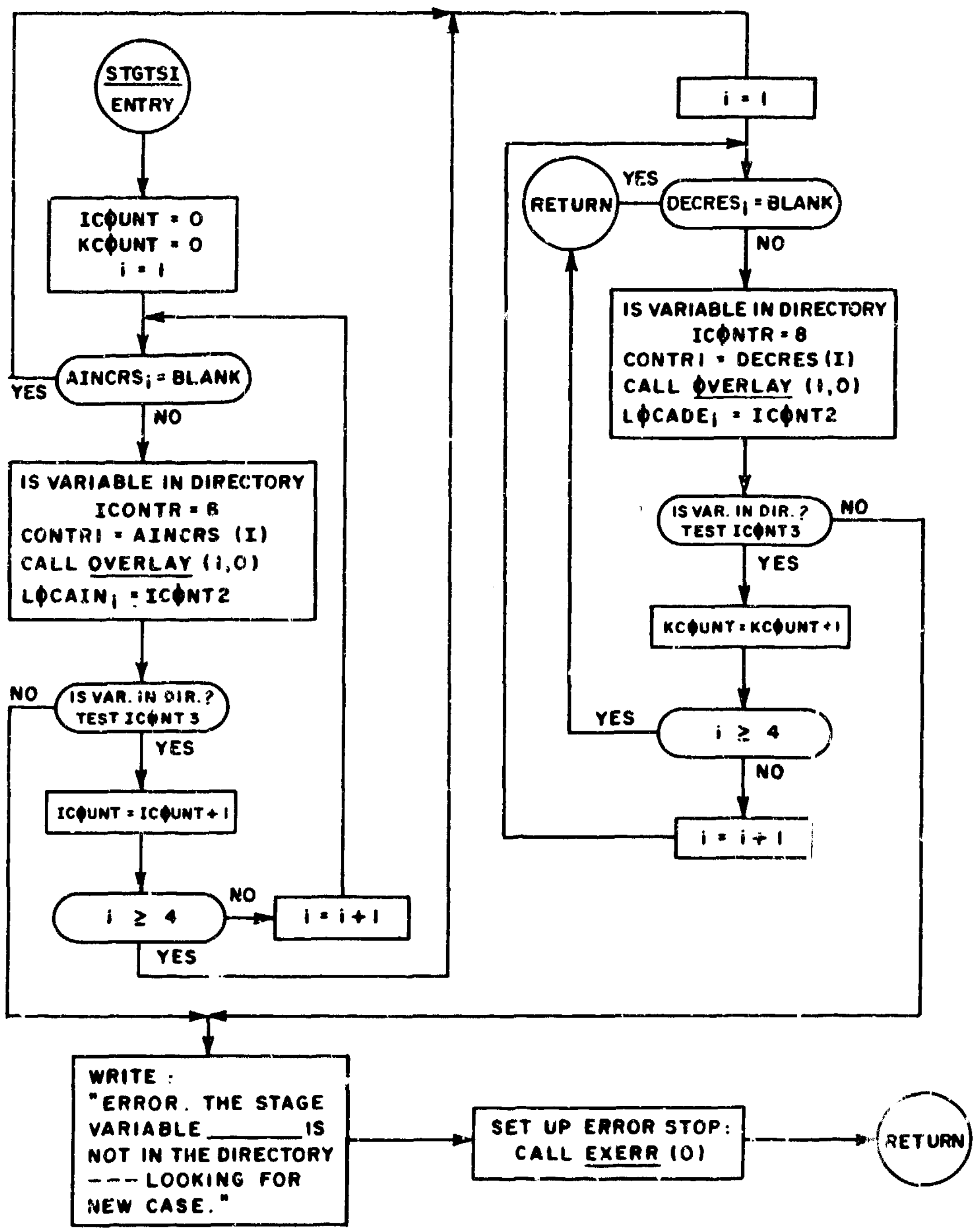
(2) Testing Staging - To test if staging is indicated,

CALL STGTST (INDSTG)

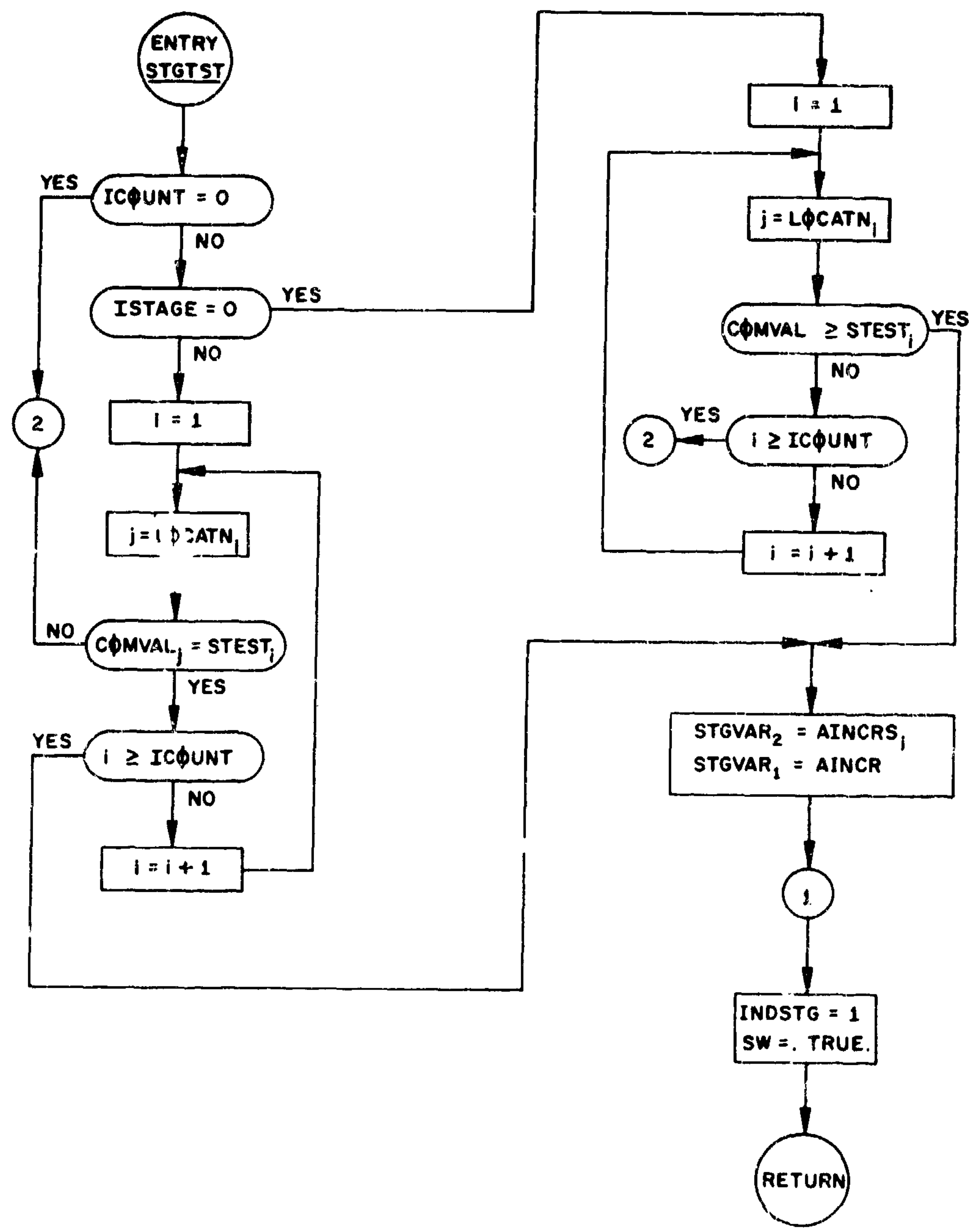
The routine returns INDSTG = 0, No staging

INDSTG = 1, staging indicated

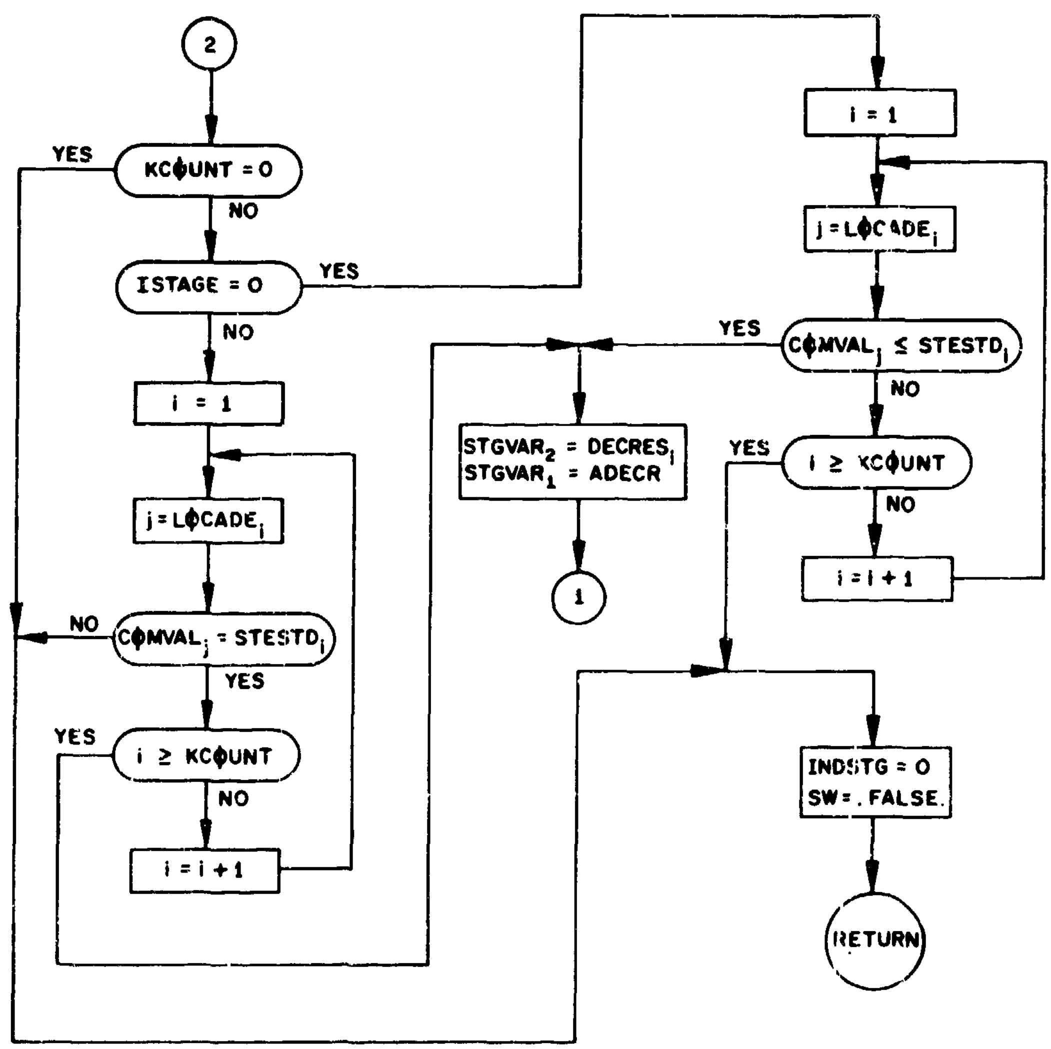
FLOW DIAGRAM (STGTSI)



FLOW DIAGRAM (STGTST)



FLOW DIAGRAM (STGTST)



4. INUPD, LNUPD, INPUZ, INTEG, UPDATE - INTERFACE ROUTINES FOR INTEGRATION ROUTINE

a. Purpose - To serve as an interface between the integration routine proper (MIMIN) and any routine requesting a variable to be integrated.

There are five logical functions which these routines perform. For a particular call to one of these routines, one of these functions will be enacted. The P array is the array of current derivative values of the variables that are being integrated. The Y array is the array of the current integrated variable values.

b. Usage - for each of the interface routines.

(1) CALL INUPD(N,L)

The number of integrated variables is increased by N. The subscripts in the P and Y arrays for the values XDOT and X respectively are stored in the array L.

(2) CALL LNUPD (M)

The number of integrated variables is decreased by M.

(3) CALL INPUZ

The P and Y arrays are set to 0; the number of integrated variables is set to 0.

(4) CALL INTEG (K,XDOT)

The value of XDOT is stored in P(K).

(5) Call UPDAT (JX1, JX2, XJ1, XJ2, XJ3, XJ4, XJ5)

JX1 = No. of variables ($1 \leq JX1 \leq 5$); JX2 = subscript of first variable (XJ1); 2nd variable (XJ2) has subscript of JX2+1, etc.

When the logical variable SKIPUP (DIRCOM) is false, then the values of the integrated variables are picked up from the proper places in the Y array and put in XJ1, XJ2, etc.

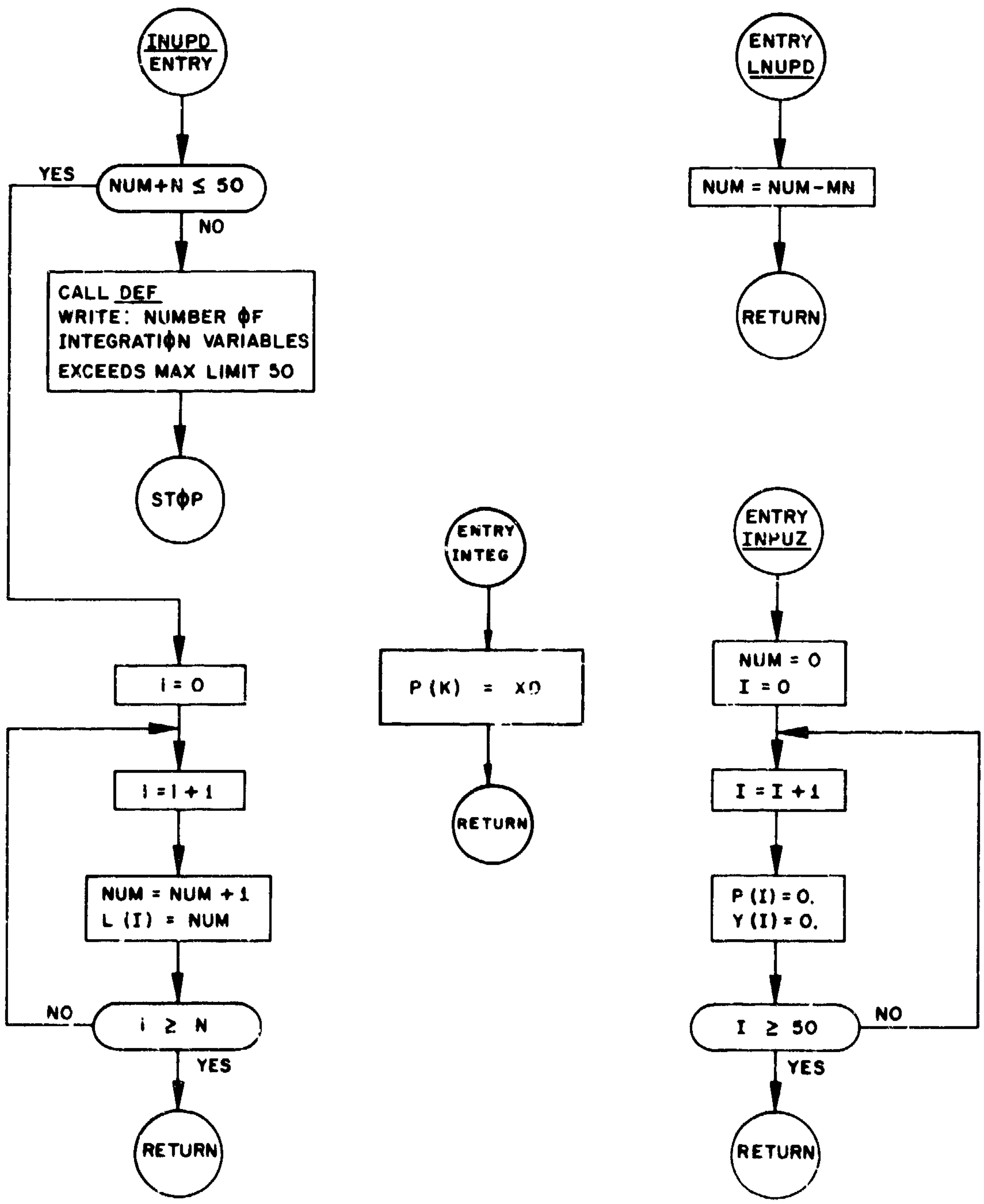
When SKIPUP is true, the values of XJ1, XJ2, . . ., XJ5 are stored in Y (JX2), Y (JX2+1), . . ., Y (JX2+JX1 - 1).

INUPD will terminate the case if more variables are requested to be integrated than there is room for in the integration arrays; at the present, this upper limit is 50 variables.

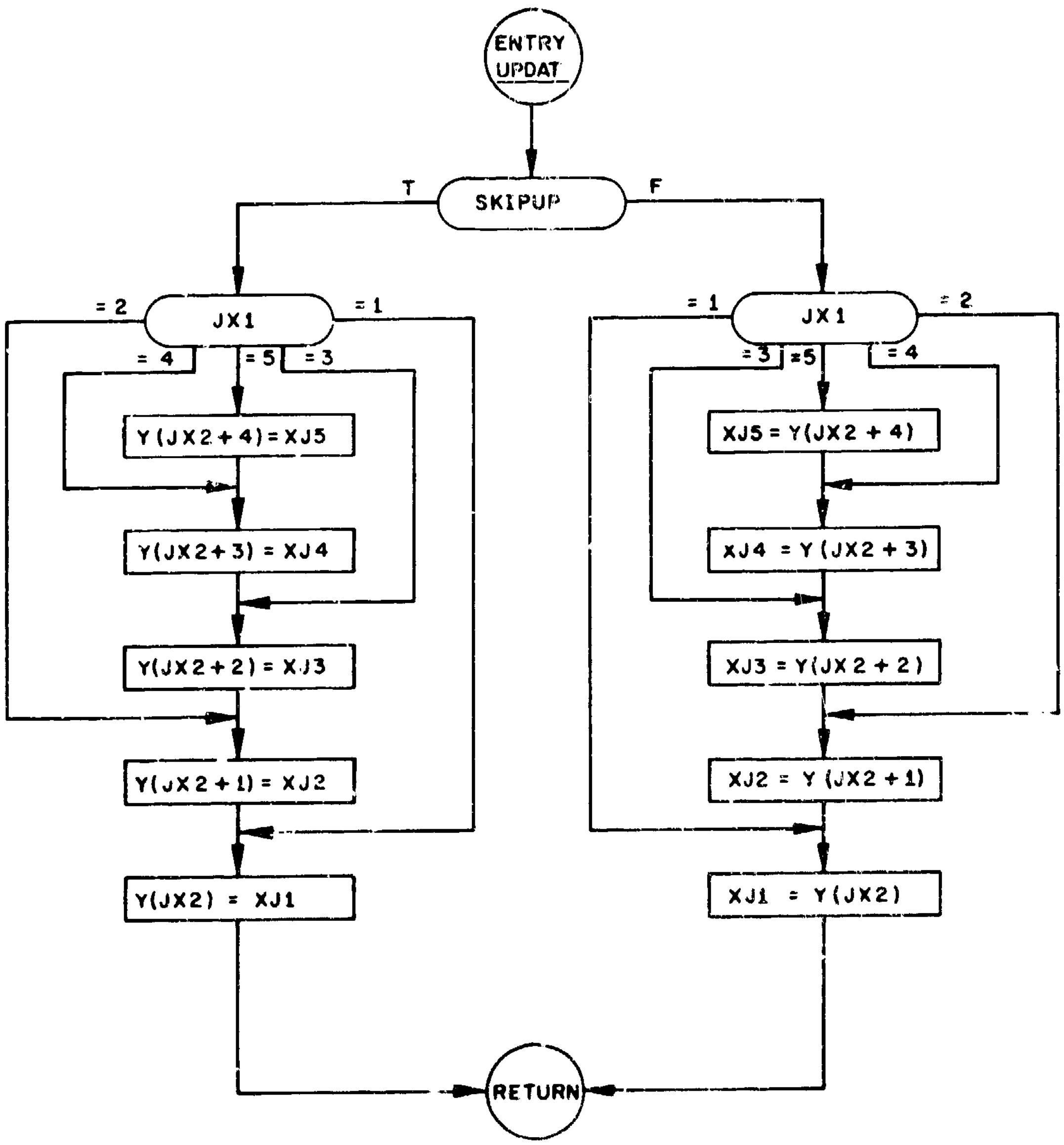
5. MIMIN - INTEGRATION ROUTINE

a. Purpose - To perform the calculation necessary to integrate an array of variables by the variable-step or fixed step Runge-Kutta Method; to determine an estimate of the relative error, and from this information, determine the new step size of integration in the variable-step mode of integration.

FLOW DIAGRAMS, (INUPD, LNUPD, INPUZ, INTEG)



FLOW DIAGRAM (UPDAT)



b. Usage - The entry to this routine is as follows:

CALL MIMIN (SN)

where N = statement number to which nonstandard return is made from MIMIN.

c. Method - The variable step Runge-Kutta Method is calculated as follows:

STEP I

$$
\begin{aligned}
Y_{max_n} &= |Y_n| \\
\dot{Y}_n &= F(X, Y_n) \\
Y_{An} &= Y_n + h/2\ \dot{Y}_n \\
\dot{Y}_{An} &= F(X + h/2, Y_{An}) \\
Y_{Bn} &= Y_n + h/2\ \dot{Y}_{An} \\
\dot{Y}_{Bn} &= F(X + h/2, Y_{Bn}) \\
Y_{Cn} &= Y_n + h\ \dot{Y}_{Bn} \\
\dot{Y}_{Cn} &= F(X + h, Y_{Cn}) \\
Y_{Pn} &= Y_n + h/6\,(\dot{Y}_n + 2\ \dot{Y}_{An} + 2\ \dot{Y}_{Bn} + \dot{Y}_{Cn})
\end{aligned}
$$

STEP II

$$
\begin{aligned}
Y_{Dn} &= Y_n + h/4\ \dot{Y}_n \\
\dot{Y}_{Dn} &= F(X + h/4, Y_{Dn}) \\
Y_{En} &= Y_n + h/4\ \dot{Y}_{Dn} \\
\dot{Y}_{En} &= F(X + h/4, Y_{En}) \\
Y_{Fn} &= Y_n + h/2\ \dot{Y}_{En} \\
\dot{Y}_{Fn} &= F(X + h/2, Y_{Fn}) \\
Y_{1n} &= Y_n + h/2 \cdot 1/6\,(\dot{Y}_n + 2\ \dot{Y}_{Dn} + 2\ \dot{Y}_{En} + \dot{Y}_{Fn})
\end{aligned}
$$

STEP III

$$
\begin{aligned}
Y_{Gn} &= Y_{1n} + h/4 \ \dot{Y}_{1n} \\
\dot{Y}_{Gn} &= F(X + 3/4\,h, Y_{Gn}) \\
Y_{Hn} &= Y_{1n} + h/4 \ \dot{Y}_{Gn} \\
\dot{Y}_{Hn} &= F(X + 3/4\,h, Y_{Hn}) \\
Y_{In} &= Y_{1n} + h/2 \ \dot{Y}_{Hn} \\
\dot{Y}_{In} &= F(X + h, \dot{Y}_{In}) \\
Y_n &= Y_{1n} + h/2 - 1/6\,(\dot{Y}_{1n} + 2\,\dot{Y}_{Gn} + 2\,\dot{Y}_{Hn} + \dot{Y}_{In})
\end{aligned}
$$

where n = 1, 2, 3, ..., NDEFEQ

NDEFEQ = No. of differential equations

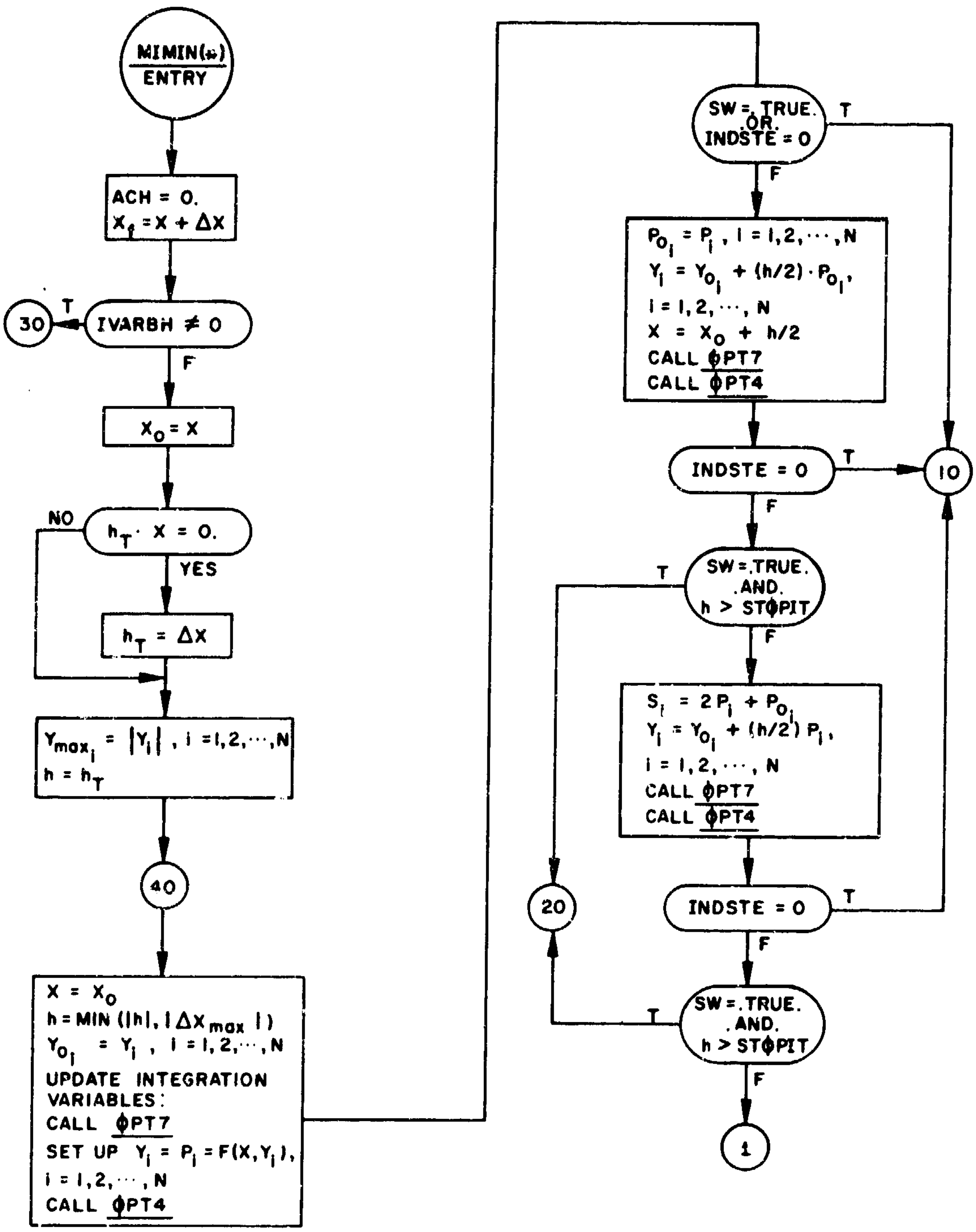
Computation of Relative Error $E_R = \frac{1}{15}(Y_n - Y_{Pn})$

$$Z_n = Max\left(Y_{Max_n}, \left|Y_n\right|\right); \ R = Max\left[R, \left|E_R\right| / Max\left(Z_n, 1.\right)\right]$$

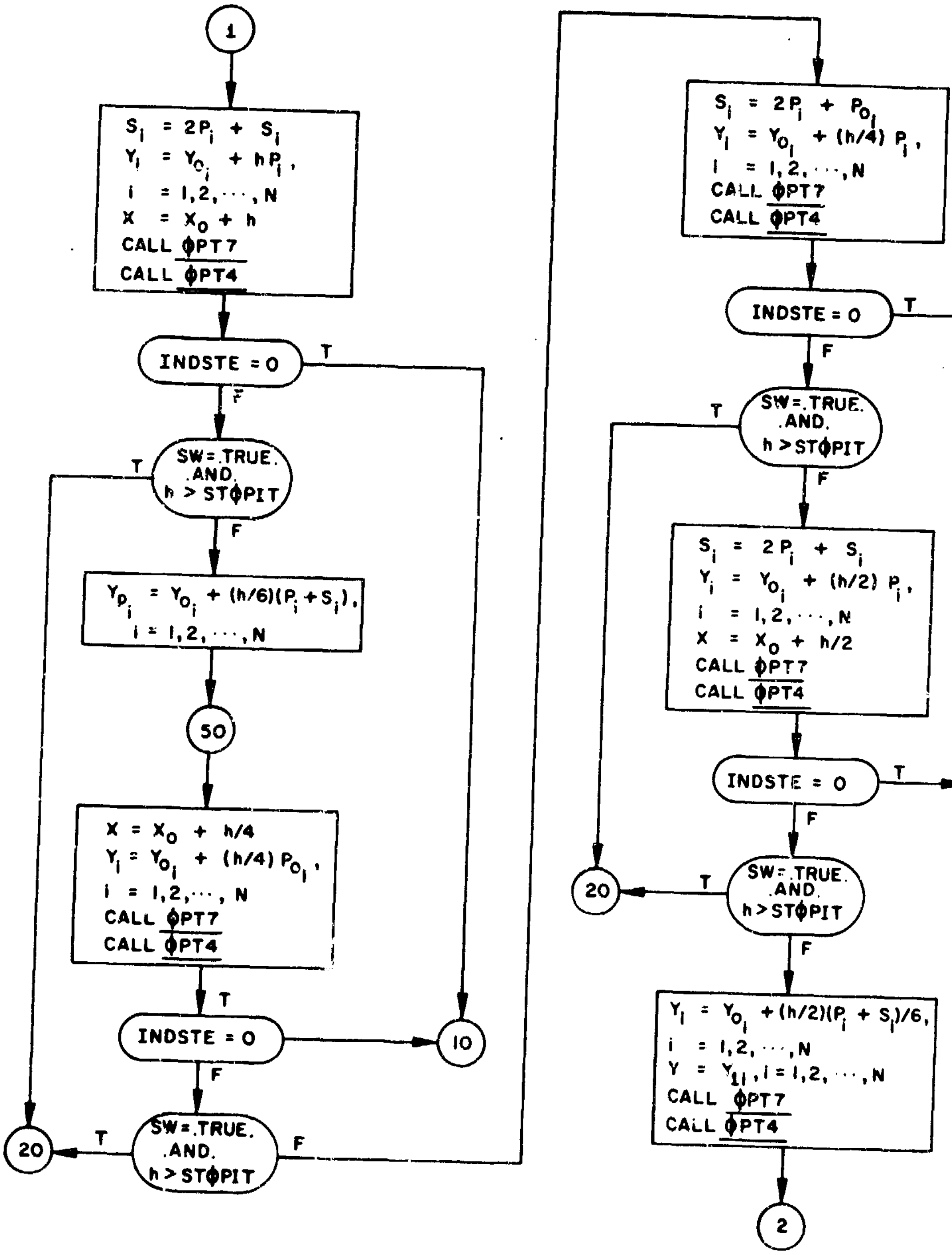
d. Input for Integration Routine

Symbol Used by READ Routine	Math Notation	Symbol used by Integration	Nominal Value	Remarks
IVARBH		IVARBH	0	Use variable-step =1, Use Fixed Step
TIME	t	X	0.	Time to begin integ.
DELTS	Δt	DX	.1	Time internal to int.
AMINER	Δt_{min}	DXMIN	.001	Minimum Δt
AMAXER	Δt_{max}	DXMAX	10000.	Maximum Δt
RELER1	$R_{EL_{FR1}}$	RELER1	.00007	Rel. error tol. #1
RELER2	$R_{EL_{ER2}}$	RELER2	.000005	Rel. error tol. #2
	N	N		No. of diff. eqs.
	Y_i	Y(N)		Array of dep. -var.
	P_i	P(N)		Array of
PRTMIN		PRTMIN		Print Minimum
		INDLG		Landing gear indicator
		INDSTE	1	
		STØPIT		
		SW		

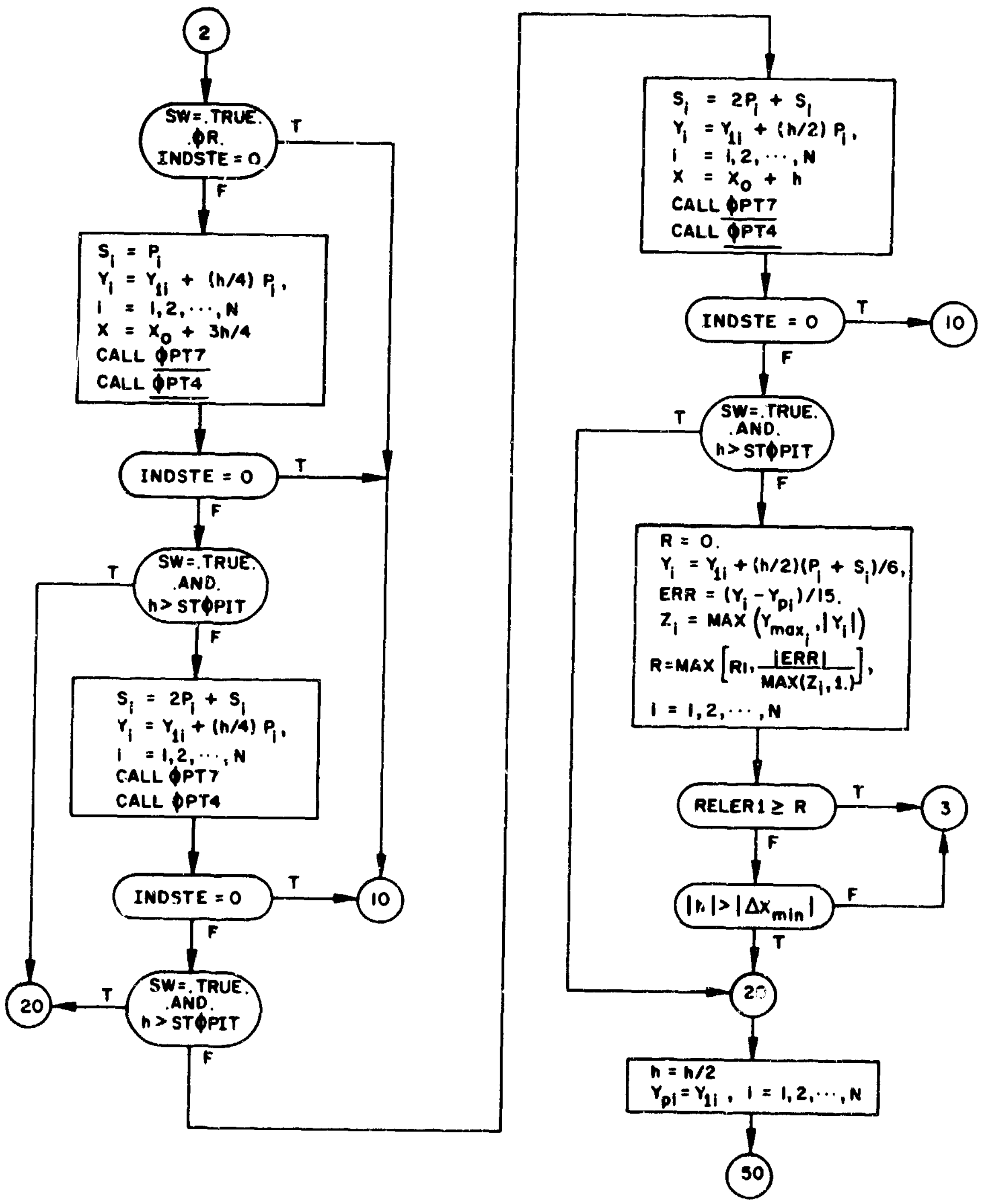
FLOW DIAGRAM (MIMIN)



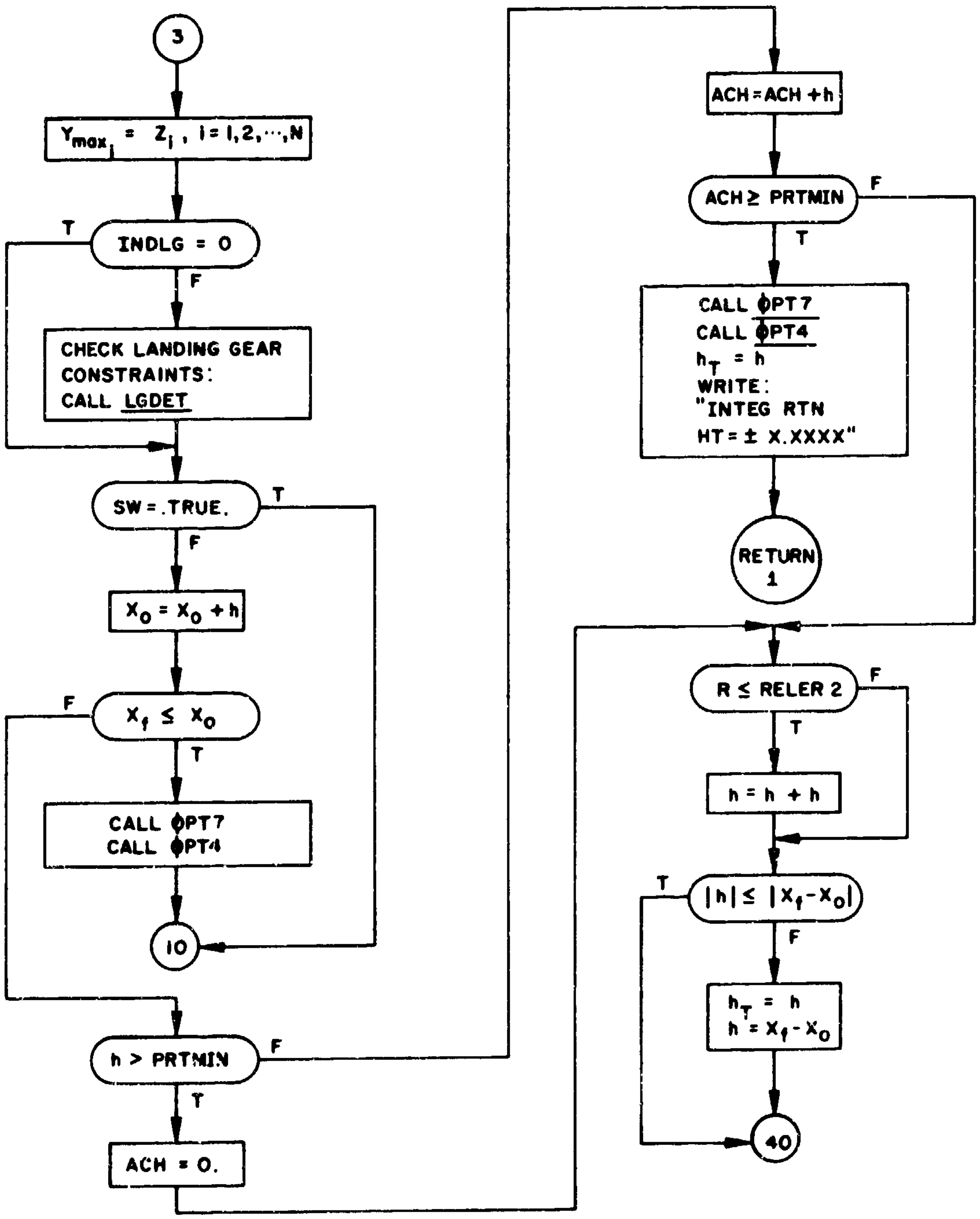
FLOW DIAGRAM (MIMIN)



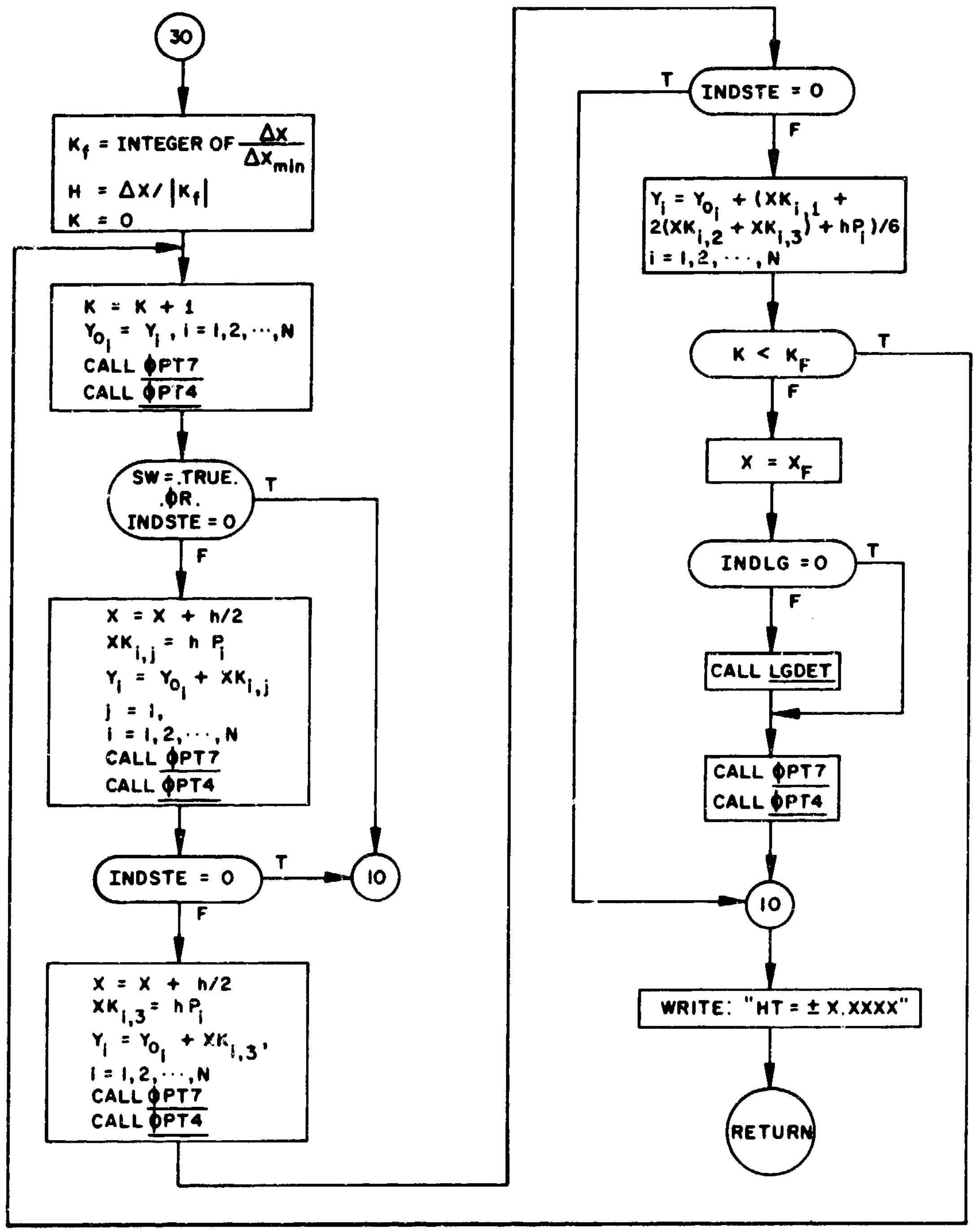
FLOW DIAGRAM (MIMIN)



FLOW DIAGRAM (MIMIN)



FLOW DIAGRAM (MIMIN)



6. LGDET - ROUTINE TO RESTRICT LG VARIABLES IN INTEGRATION ROUTINE

a. Purpose

When the statement CALL INTEG (K, XD) is executed, the derivative XD is stored in an array P at location K of that array (i.e., P(K)). The integration is not processed until a complete pass has been made through the program and all calls to INTEG have been made.

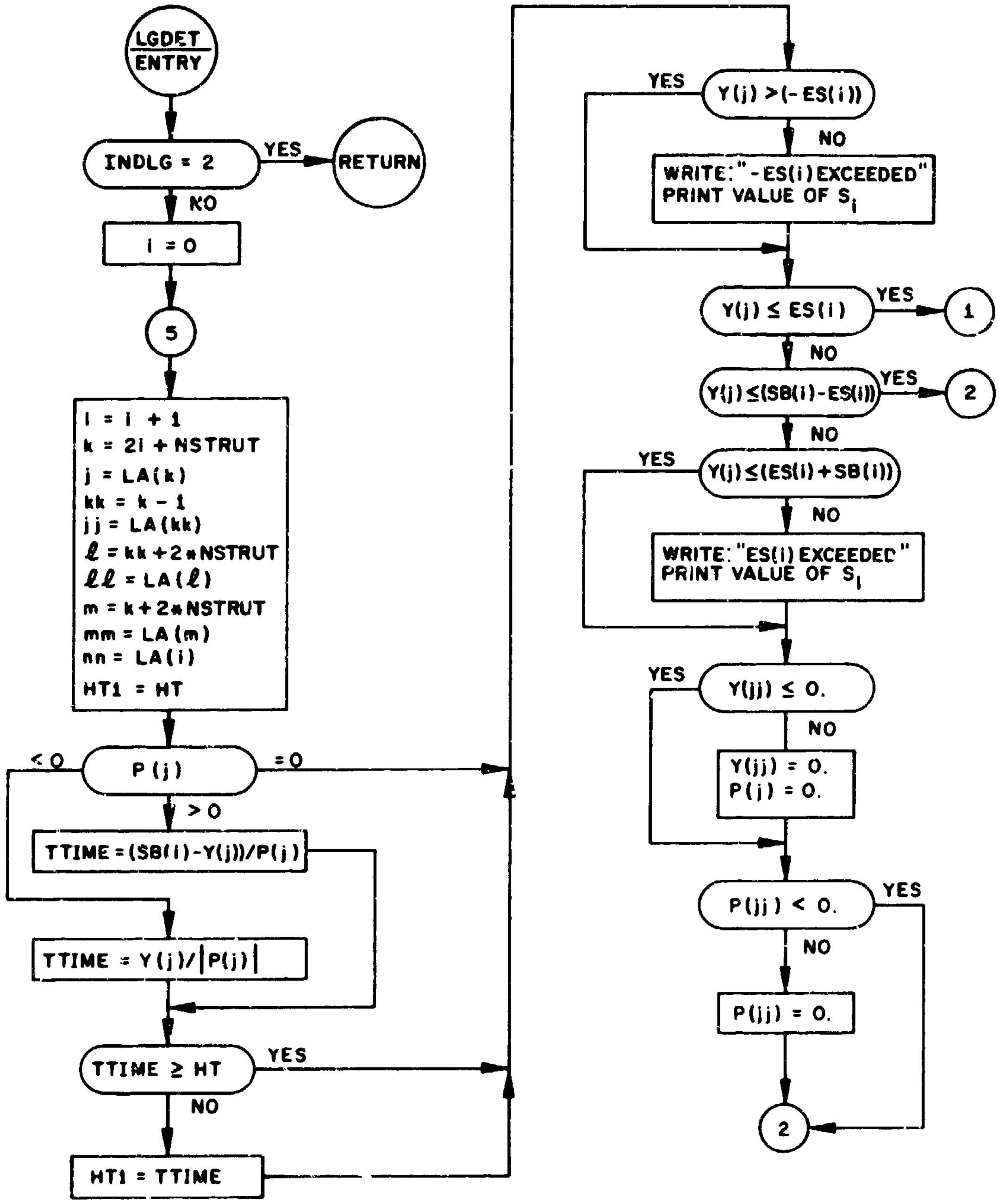
As the integration is performed the integrated variables are stored in an array Y at the corresponding location K as its derivative (i.e., Y (K)).

When the statement CALL UPDAT (N,K,X) is executed, the integrated variable X is transmitted from the Y array (location Y (K)) to the location X. The N designates the number of variables to be transmitted.

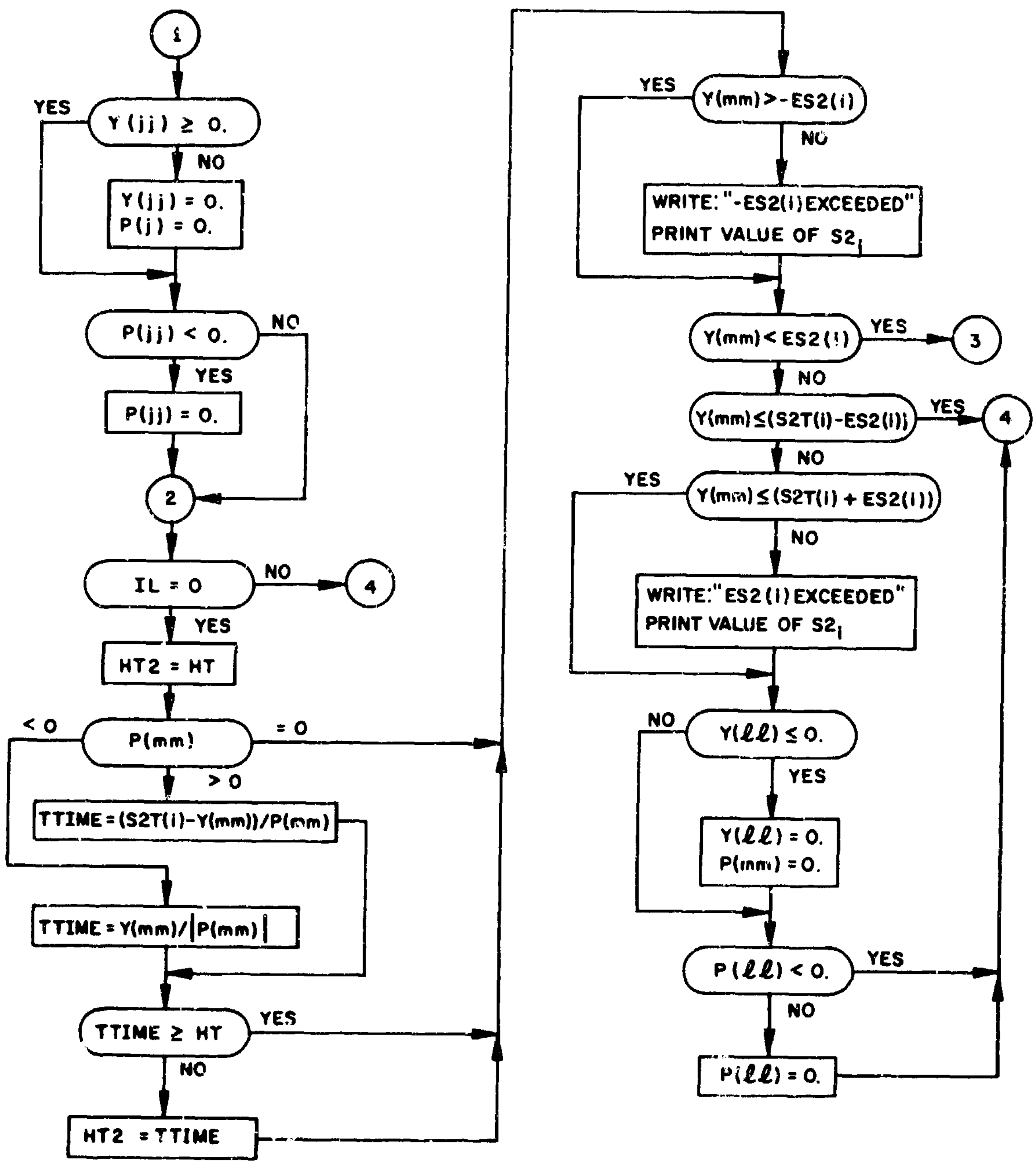
Due to the requirements of the landing gear problem some variables that are integrated are restricted to certain conditions. Therefore, it was necessary to write the routine LGDET to restrict these variables in the integration routine. As stated above these variables are stored in the P and Y array in the integration routine.

b. Linkage - CALL LGDET

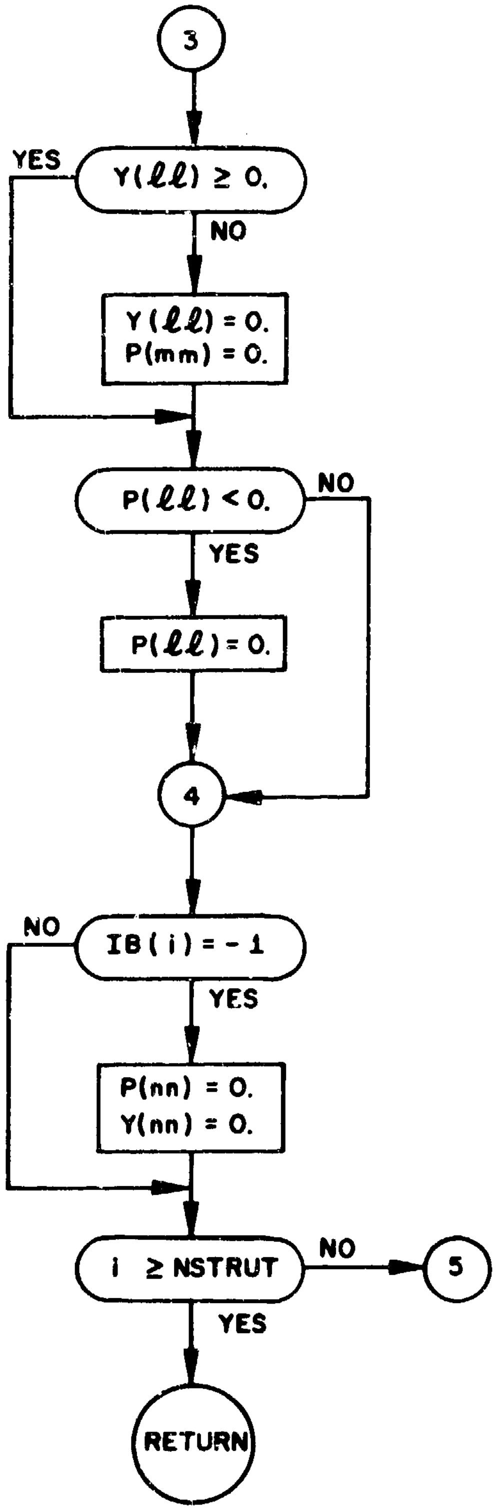
FLOW DIAGRAM (LGDET)



FLOW DIAGRAM (LGDET)



FLOW DIAGRAM (LGDET.)



7. ASRCH, TDATA - DIRECTORY SEARCH ROUTINES.

a. Purpose - To provide a BCD word look-up from a subscript.

b. Method - Given a subscript, the routine will search the directory for the BCD word corresponding to that subscript. If the subscripts do not compare the BCD name is set to blank and return to the Calling Program is made.

c. Usage - Entry is made to the routine with the following statement:

a. CALL ASRCH (LOC1, SYM1)

where

LOC1 = subscript being searched

SYM1 = the variable name into which the routine is to store the corresponding BCD name.

d. A call to the subroutine TDATA initializes table name data which is used by subroutine TABRE (in Overlay (1,0)).

8. DEF - HEADING AND PAGE EJECT ROUTINE

a. Purpose - To provide page ejection and title printing.

b. Method - Initially the current page number (NPAGE) is incremented by 1. The title is printed and returned to the calling program.

c. Usage - Entry is made via the statement

CALL DEF

9. STFL, STFLD, STØVAR, ARRAY - Storage and Printing of Output Routines

a. Purpose - To provide a method of printing output names of variables and their values when necessary. Names of variables or values which are to be printed are not actually printed by the routines until at least eight have been accumulated by a series of calling sequences.

b. Method - Each time a call is made, names of variables or values are saved until eight are stored. At this time they are printed. This process is repeated until all names or values have been handled. If less than 8 remain, they are saved for further call statements or until forced to be printed.

c. Usage

(1) The printing of Hollerith Code

CALL STFL (JOPT, N, ARG1)

JOPT = 0: Force any possible remaining print

JOPT = 1: See (2) below for printing of Values of Variables.

JOPT = 2: Prints N words of Hollerith information from ARG1 (1), ARG1 (2), . . ., ARG1 (N) where there is at most 6 characters per word.

JOPT = 3: Prints 1 word of Hollerith information.

N should equal 1.

The above call adds the N Hollerith code words to the list of code words to be output on the next line of print. When 8 code words have been accumulated, the line is printed; any excess code words are added to the list for the next line of print. Codes are printed with "7X,A6,7(9X,A6)" format.

(2) The printing of Values of Variables

CALL STOVAR (N, A1, A2, . . ., A8) $1 \leq N \leq 8$

where N is an integer identifying the number of arguments following it. If $N \leq 8$, then the remaining arguments must be dummy arguments. For example, CALL STOVAR (6, A1, A2, A3, A4, A5, A6, DU, DU).

CALL STFL (JOPT, N, ARG1). This call prints the value of one variable when JOPT = 1, N = 1, and ARG1 is the name of the variable.

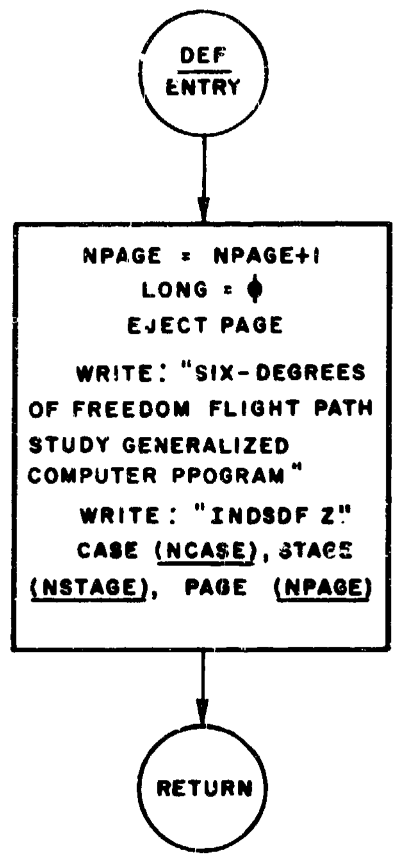
Lines accounting is taken care of within this routine. Values are printed with a "1PE15.7" format.

(3) Forcing final print

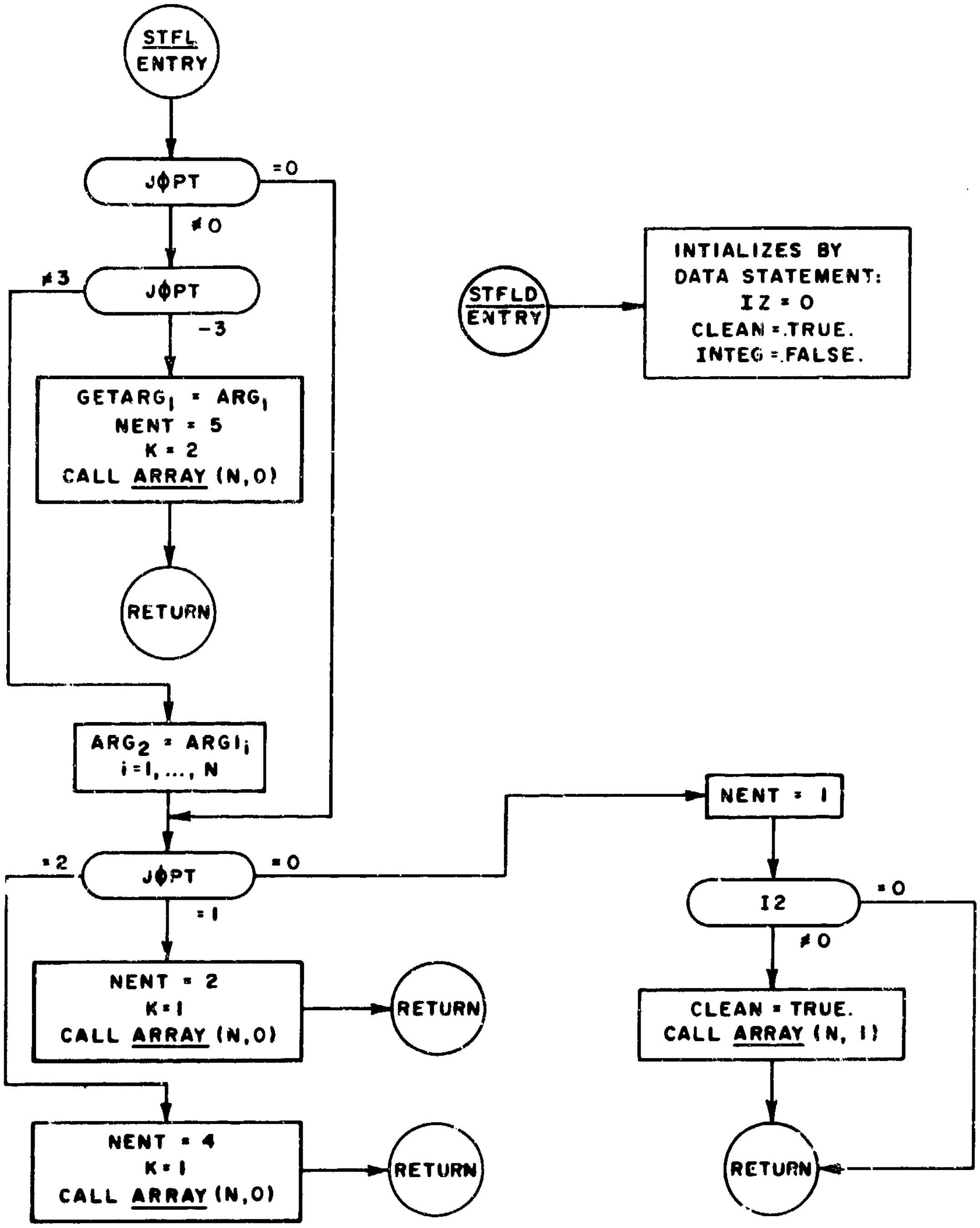
To force any possible remaining print

CALL STFL (0, 1, DU)

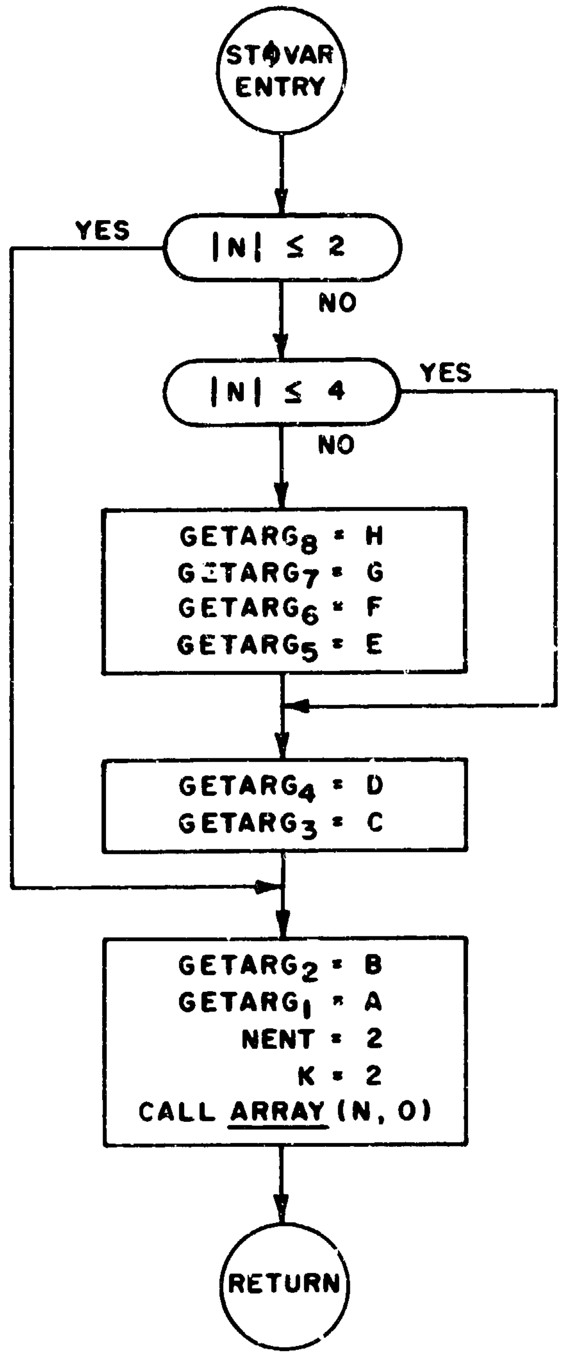
FLOW DIAGRAM (DEF.)



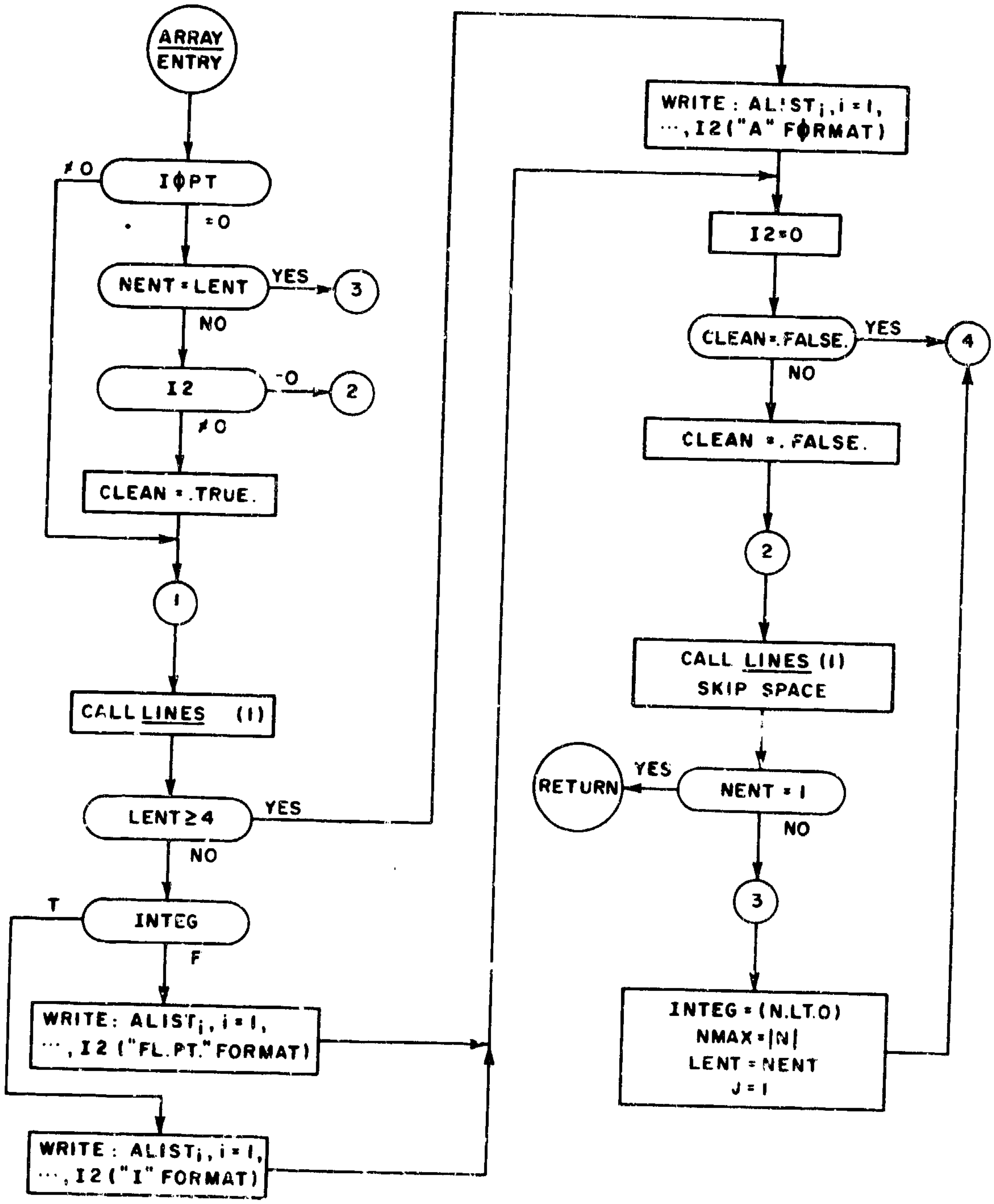
FLOW DIAGRAM (STFL)



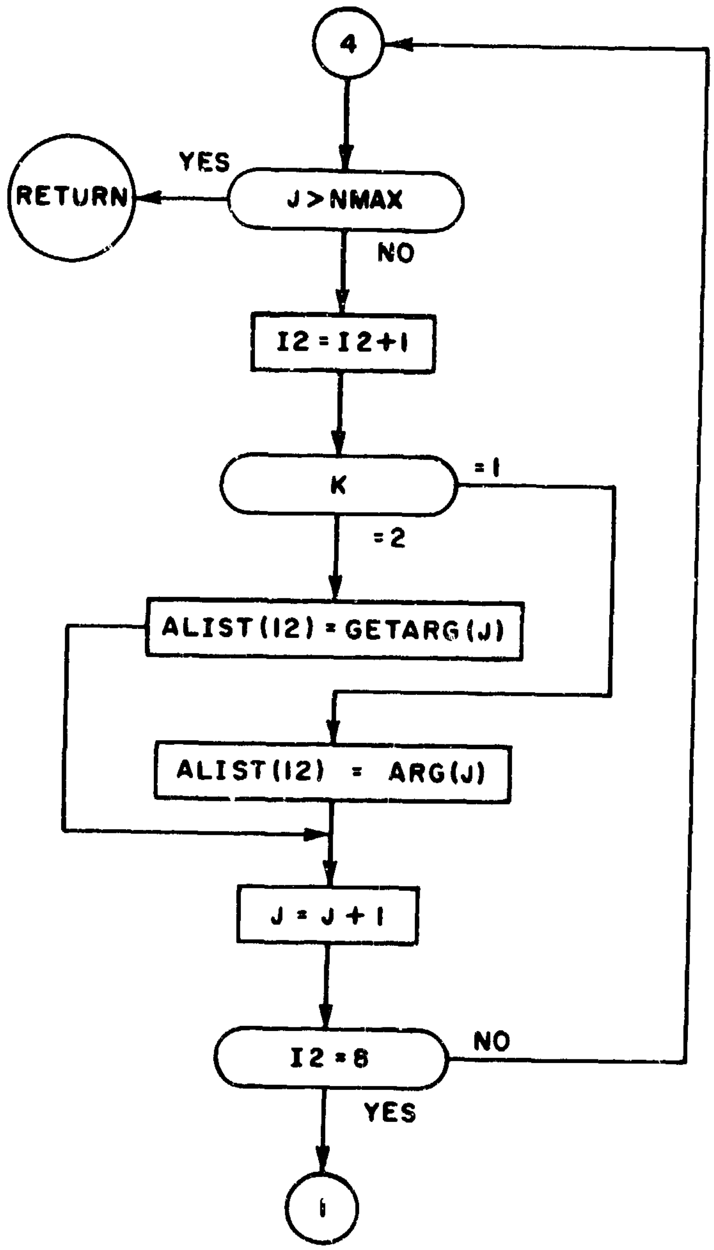
FLOW DIAGRAM (STØVAR)



FLOW DIAGRAM (ARRAY)



FLOW DIAGRAM (ARRAY)



10. LINES - LINES ACCOUNTING ROUTINE

a. Purpose - To keep an accounting of the number of lines printed per page, and to provide for page control.

b. Method - If the number of lines to be printed (LCØUNT) is such that it will not fit on the current page, the page is ejected (via DEF) and printing will begin on the new page. Initially, the location LØNG should be set to zero, indicating that currently no lines have been printed on the present page.

c. Usage - Entry is made via the statement,

CALL LINES (LCØUNT)

where

LCØUNT = A fixed point variable or constant indicating the number of lines to be printed.

11. ASIN - ARC SINE FUNCTION

a. Purpose - To compute the arc sine of a normalized floating point argument (X).

b. Method -

$$\text{Sin}^{-1}(X) = \frac{\pi}{2} - \text{Cos}^{-1}(X)$$

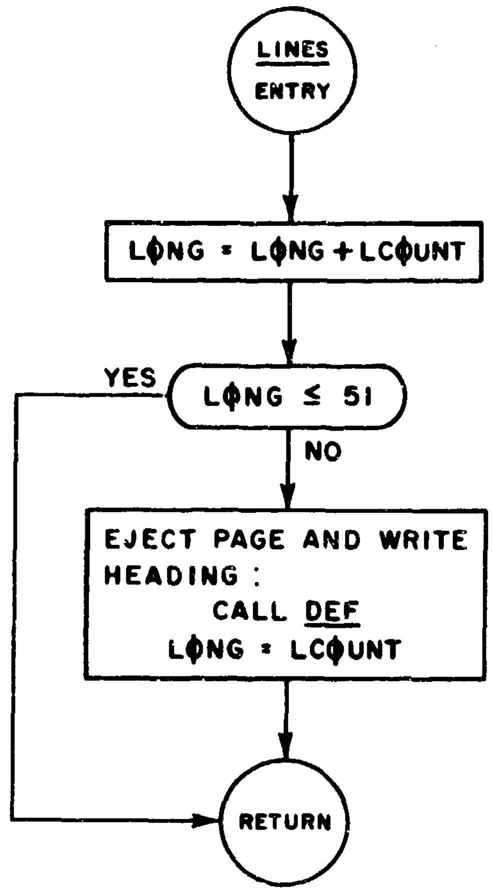
c. Usage - The Arc Sine is computed using the statement, Y = ASIN (X) where $|X| \leq 1.$, and $Y = \text{Sin}^{-1}(X)$.

12. ACOS - ARC COSINE FUNCTION

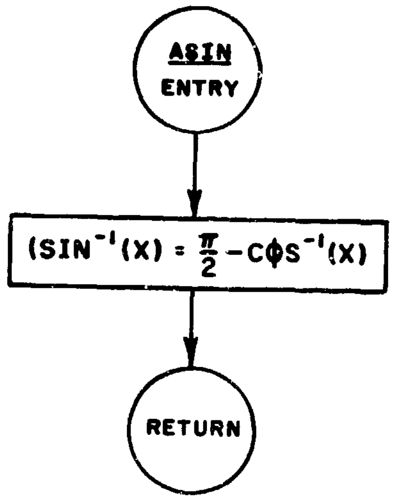
a. Purpose - To compute the arc cosine of a normalized floating point argument X.

b. Method - For $|X| < 7.4505806 \times 10^{-9}$ the arc cosine is set equal to $\pi/2$. For X = 1., arc cosine is set equal to zero, and for X = -1 arc cosine is set equal to π. When the argument $X \neq \pm 1.$, the routine gives the arc cosine in radians from 0 to π.

FLOW DIAGRAM (LINES)



FLOW DIAGRAM (ASIN)



c. Usage - The arc cosine is computed using the statement

$$Y = AC\text{Ø}S\ (X)$$

where $|X| \leq 1.$ and $Y = \text{Cos}^{-1}\ (X)$.

13. ATAN2 - ARC TANGENT ROUTINE

a. Purpose - To compute the arctangent of the quotient of two normalized floating point quantities Y/X, with proper quadrant control.

b. Method - The routine computes the quotient Y/X. The arctangent is computed with quadrant according to the sign of Y and X. If X = 0 and $Y \neq 0$, the routine computes $Z = \text{Tan}^{-1}\ (Y/X) = \frac{Y}{|Y|} \cdot \frac{\pi}{2}$.

If X = 0 and Y = 0, it computes

$$Z = \text{Tan}^{-1}\ (Y/X) = 0.$$

c. Usage - The arctangent of Y/X is obtained via

```
Z=ATAN2(Y,X)
```

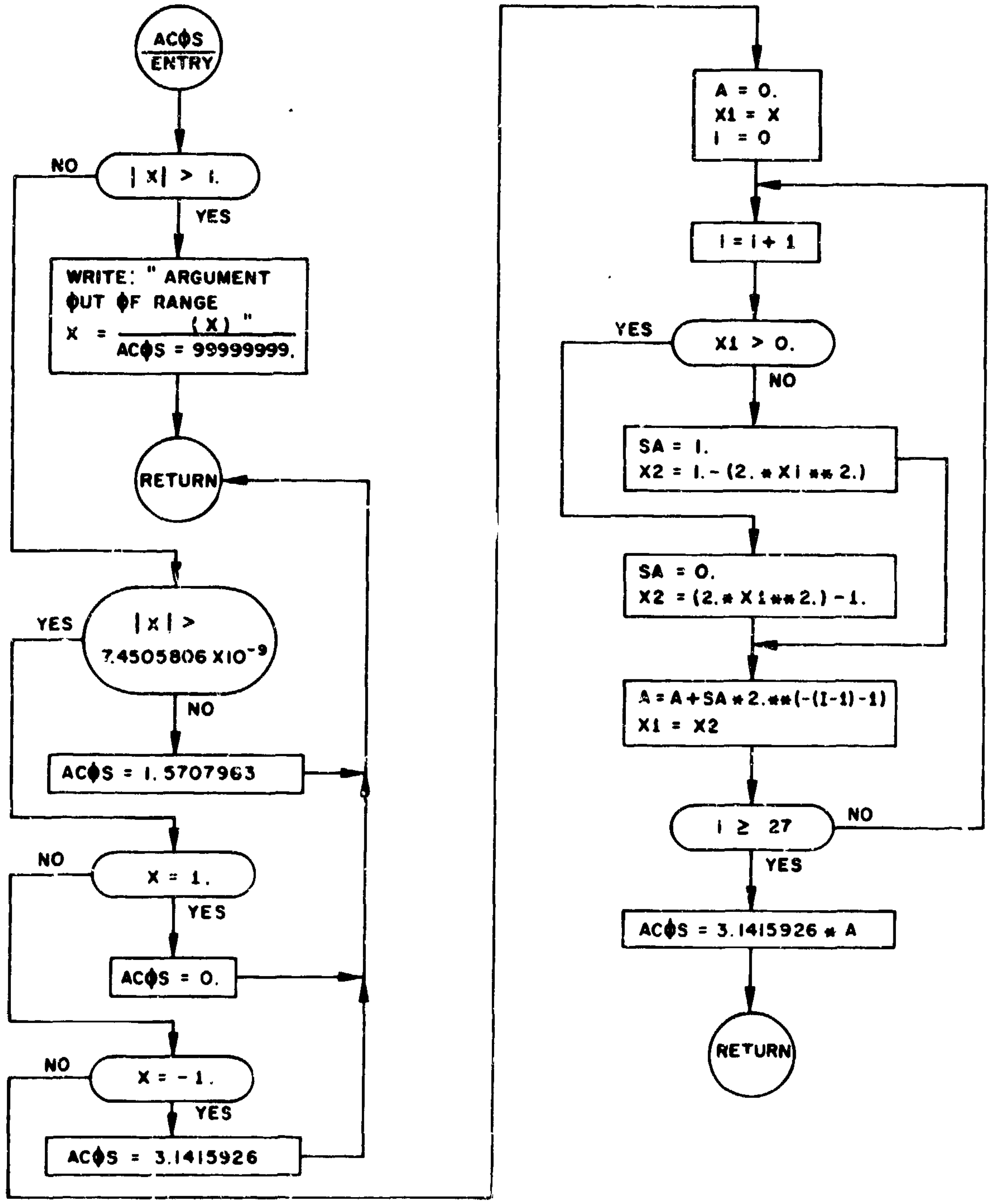
14. ERROR, EXERR - GENERAL TABLE ERROR ROUTINE

a. Purpose - To provide a method of indicating the table which may possibly contain an error. Also, to provide the Stop Number that identifies an error condition in an equation.

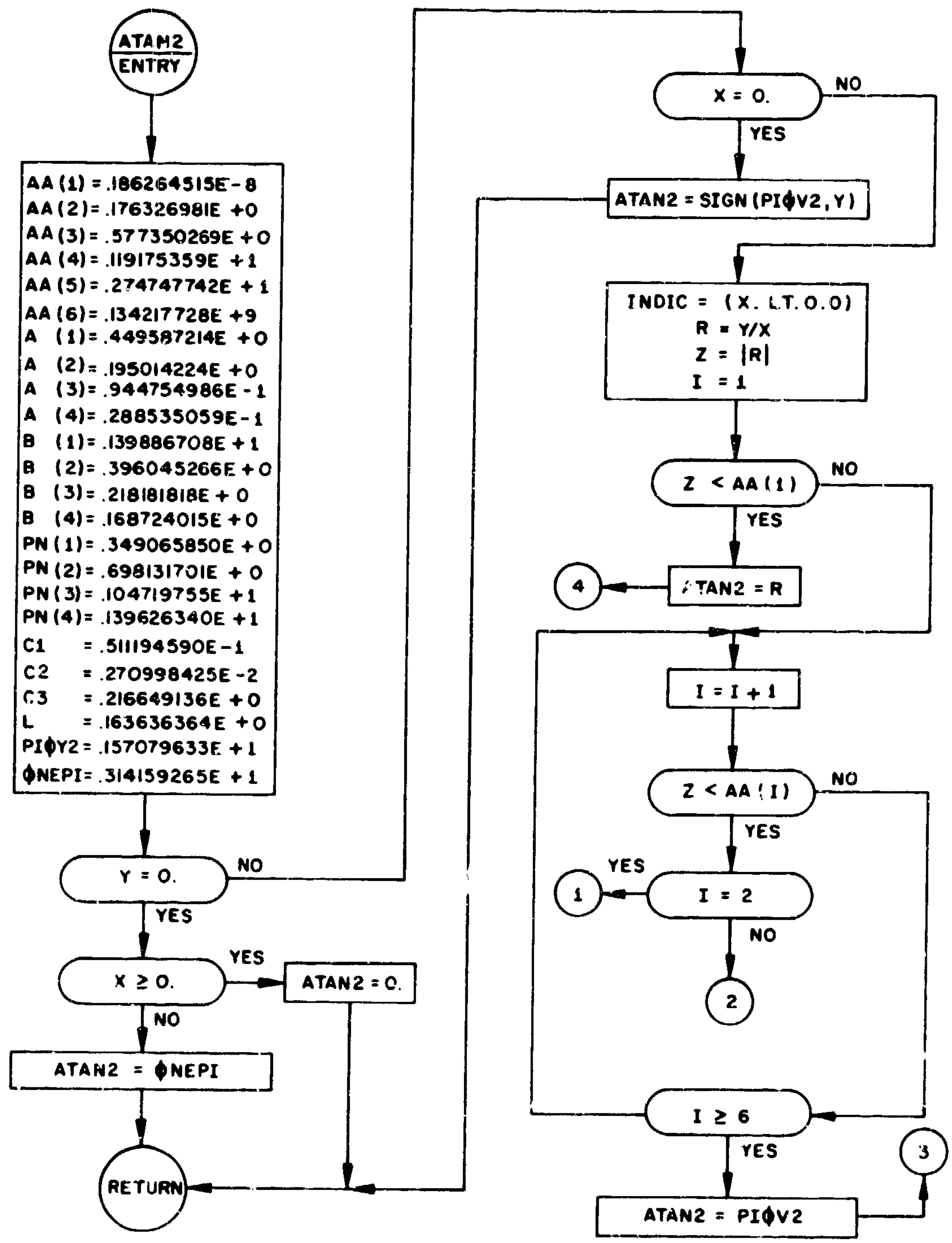
b. Method - By the statement CALL ERROR (LOCT) in which LOCT is the subscript of the curve in error, the routine will search the subscript table and find the corresponding BCD word. This word will then be printed as:

"TABLE ERROR AAAAAA"

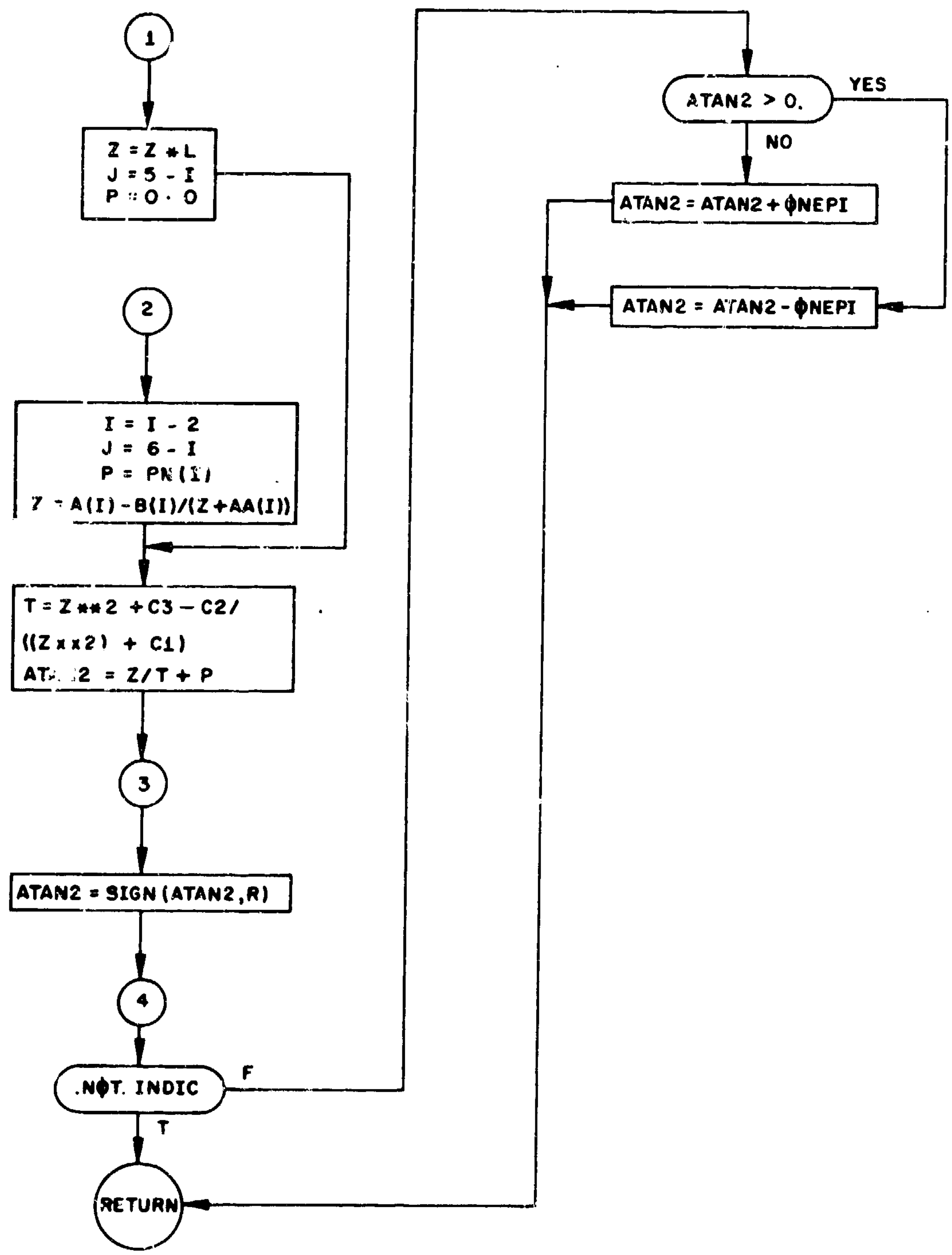
FLOW DIAGRAM (ACϕS.)



FLOW DIAGRAM (ATAN2)



FLOW DIAGRAM (ATAN2)



Where AAAAAA is the BCD name of the table. If the name cannot be found in the directory

"TABLE ERROR

LOCATION OF TABLE NOT LISTED IN DIRECTORY" is printed and a return to the calling program is made. In either case INDSTE is set to zero. By the statement CALL EXERR (NUM) in which NUM is the stop number, the routine will write "STOP NUMBER III"
where III = the stop number. If NUM = 0, an exit is made from the routine with no printing. In either case INDSTE is set to zero.

c. Usage - Entries are made to the routine with the following statement:

(1) CALL ERROR (LOCT)

where LOCT is the table subscript.

(2) CALL EXERR (NUM)

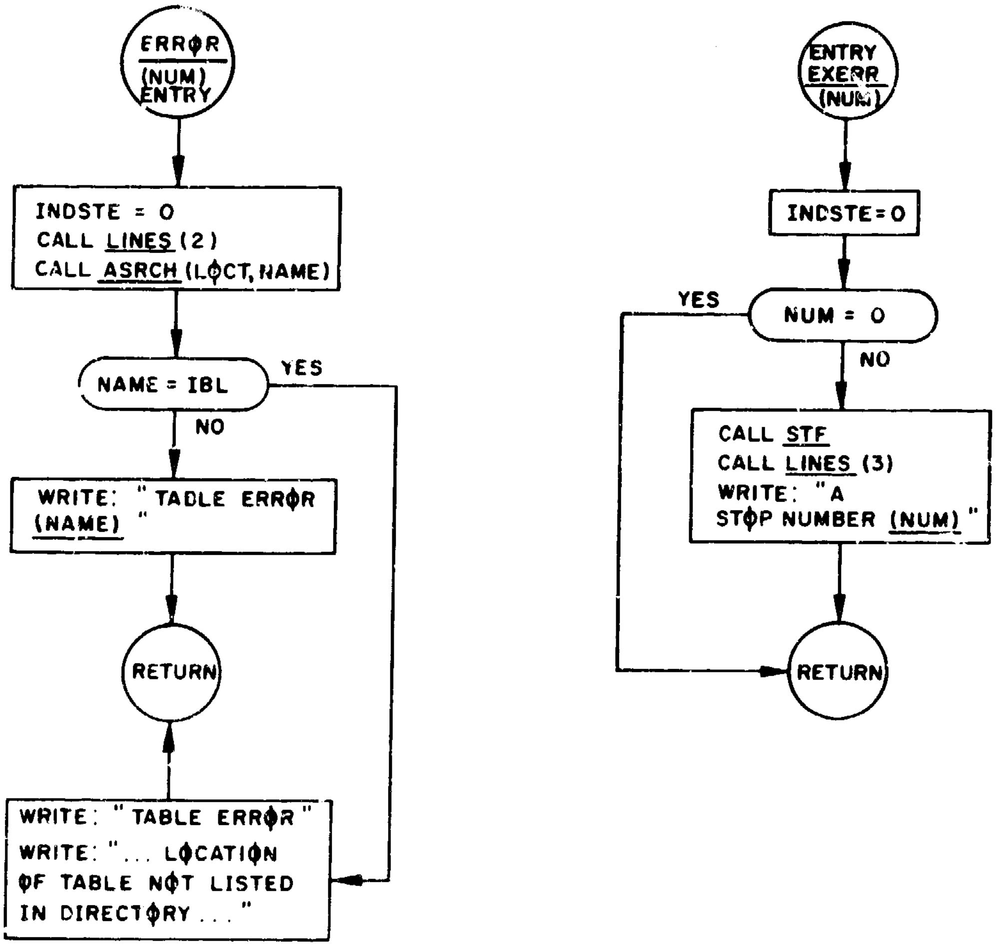
where NUM is the stop number.

15. NDTLU - N-DIMENSIONAL TABLE LOOK-UP ROUTINE

a. Purpose - To provide a method of linearly interpolating in a table of n independent variables.

b. Method - Given the arguments X(1), X(2), . . . , X(N-1), the routine computes Y = F(X(1), X(2), . . ., X(N-1)) by linear interpolation from a table.

FLOW DIAGRAM (ERRØR, EXERR)



c. Usage - Entry is made via the statement:

CALL NDTLU (ND, NA, X, Z, XA, ZR, IE)

where

ND = Dimension of look-up (when Y = F(X), ND = 2)

NA = An array of length ND-1. Numbers of values of each table of X. The tables are listed by size.

X = Tables of each X in order.

Z = Function Values. If A = F(X, Y, Z) the dependent variable array must be in the following order.

Assume NX=4, NY=3, NZ=2.

W(1) = F(X1, Y1, Z1)	W(13)= F(X1, Y1, Z2)
W(2) = F(X2, Y1, Z1)	W(14)= F(X2, Y1, Z2)
W(3) = F(X3, Y1, Z1)	W(15)= F(X3, Y1, Z2)
W(4) = F(X4, Y1, Z1)	W(16)= F(X4, Y1, Z2)
W(5) = F(X1, Y2, Z1)	W(17)= F(X1, Y2, Z2)
W(6) = F(X2, Y2, Z1)	W(18)= F(X2, Y2, Z2)
W(7) = F(X3, Y2, Z1)	W(19)= F(X3, Y2, Z2)
W(8) = F(X4, Y2, Z1)	W(20)= F(X4, Y2, Z2)
W(9) = F(X1, Y3, Z1)	W(21)= F(X1, Y3, Z2)
W(10)= F(X2, Y3, Z1)	W(22)= F(X2, Y3, Z2)
W(11)= F(X3, Y3, Z1)	W(23)= F(X3, Y3, Z2)
W(12)= F(X4, Y3, Z1)	W(24)= F(X4, Y3, Z2)

ZR = Results

IE = Error Code

= 0 No error

-1 X array too small

1 X array too large

2 array not in ascending order

Let n be number of indpendent variables, then the table is called an "(n+1) dimensional table."

$$Z = F(X_1, \ldots, X_n)$$

To use a table of dimension ≥ 3 and ≤ 5, a call to HIHØ should be made with the list of arguments in the calling sequence in the same order as the independent variables are numbered.

16. ATMS - ATMOSPHERE CALCULATION ROUTINE (1969)

a. Purpose - To compute the atmosphere characteristics: Density, speed of sound, pressure, temperature, and kinematic viscosity. All are a function of altitude.

b. Method - All atmosphere characteristics are computed using the 1969 ARDC model atmosphere. Values of the atmosphere characteristics are computed for positive altitudes. If an altitude is negative, the sea level value will be obtained.

c. Usage - Linkage is affected by

CALL ATMS (HGC7F)

where HGC7F = altitude in feet

17. INVR3 - INVERSE OF A NONSINGULAR 3X3 MATRIX

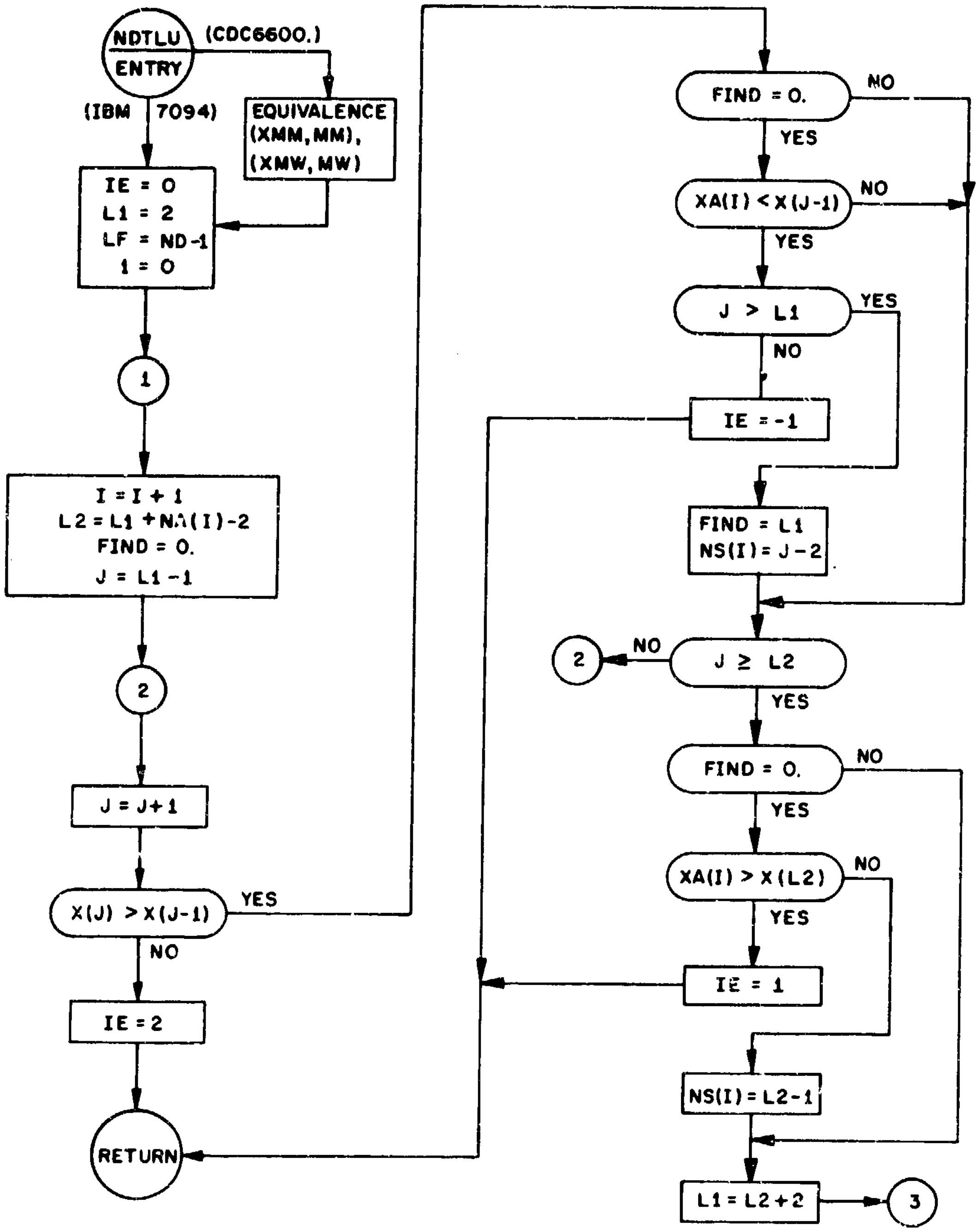
a. Purpose - To compute the inverse of a nonsingular 3 x 3 matrix.

b. Method - Let $a = [a_{ij}]$ $i,j = 1,2,3$

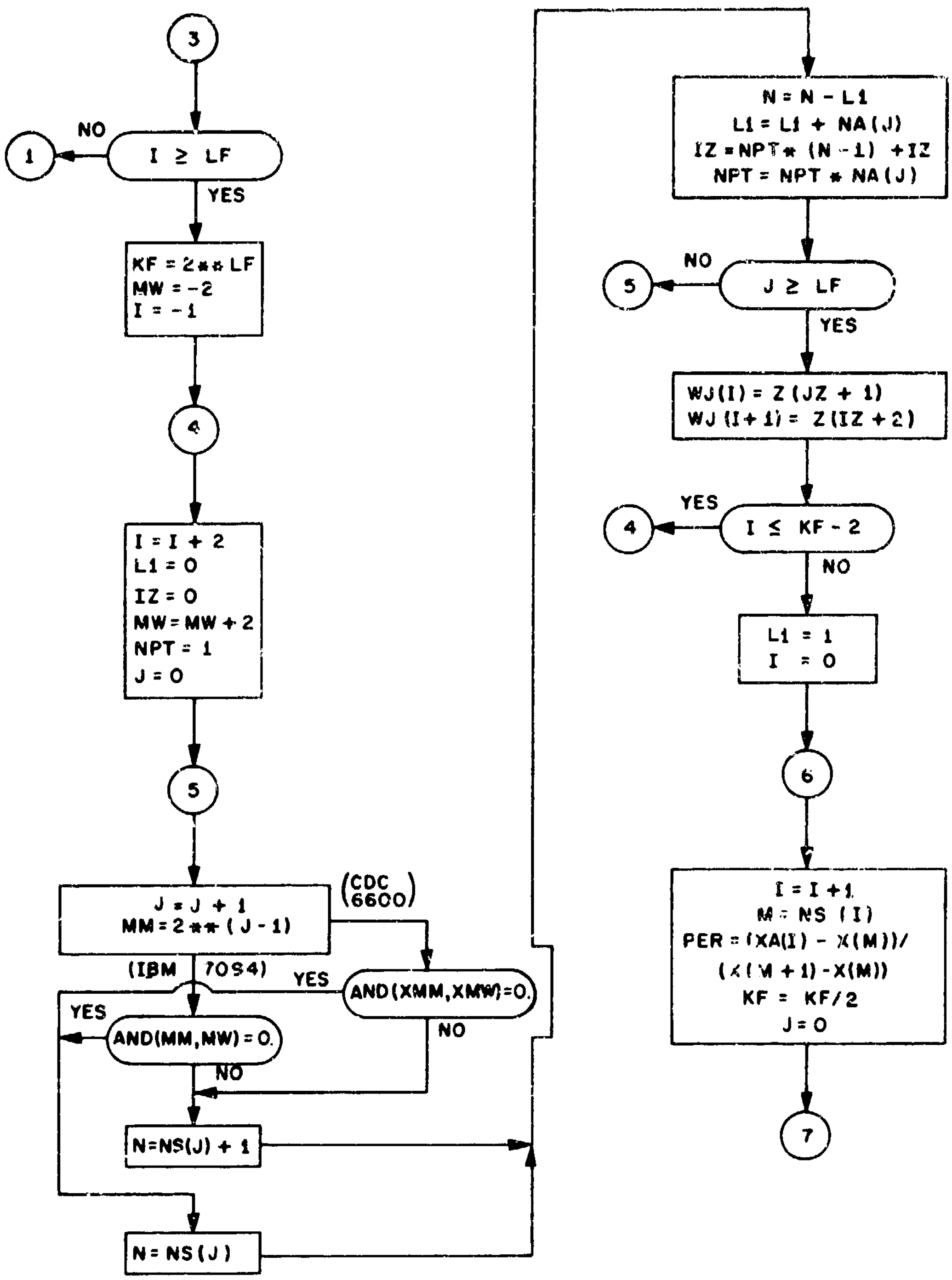
then $A^{-1} = \frac{1}{|A|} [A_{ij}]$ is computed where A_{ij} is the cofactor of a_{ij}.

The matrix A must be stored in the usual FORTRAN sense (i.e., columnwise).

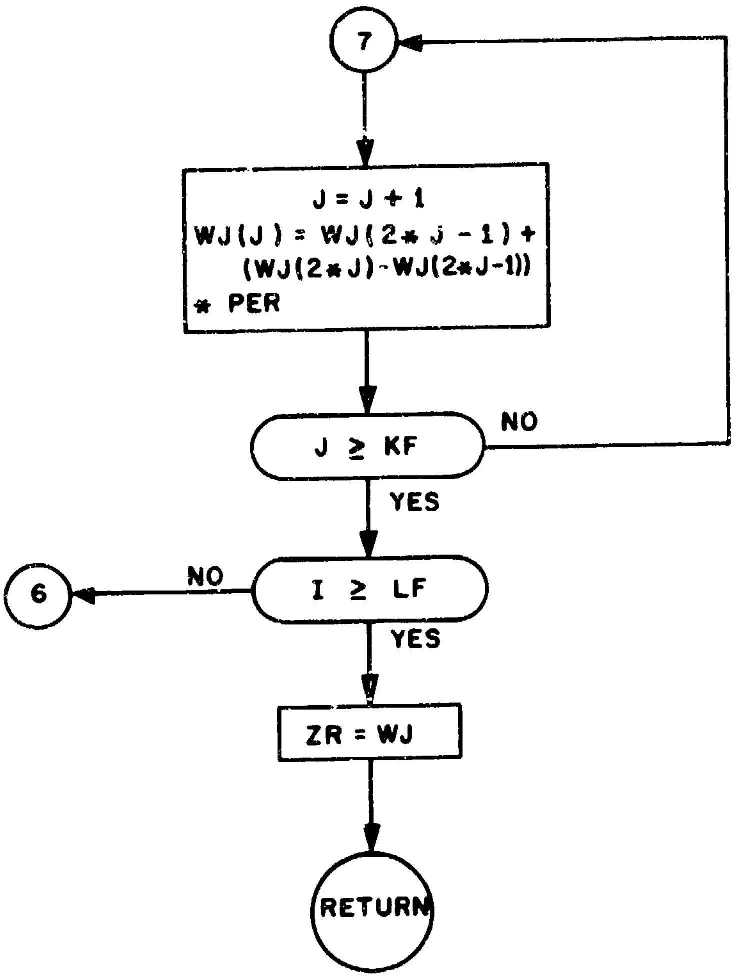
FLOW DIAGRAM (NDTLU)



FLOW DIAGRAM (NDILU)



FLOW DIAGRAM (NDTLU)



FLOW DIAGRAM (ATMS)

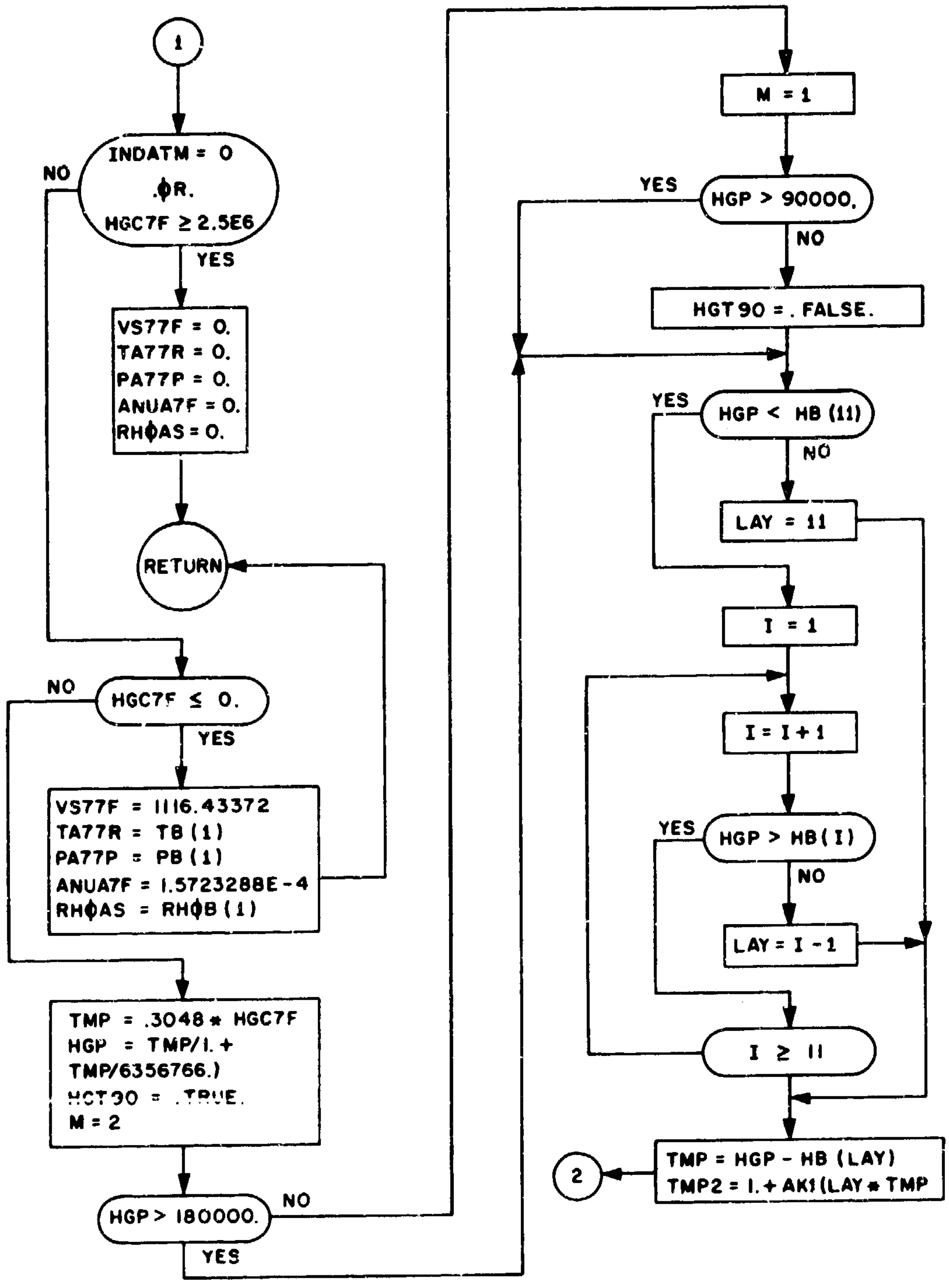
ATMS ENTRY

HB(1) = 0.
HB(2) = 11000.
HB(3) = 25000
HB(4) = 47000.
HB(5) = 53000.
HB(6) = 79000.
HB(7) = 90000.
HB(8) = 105000.
HB(9) = 160000.
HB(10) = 170000
HB(11) = 200000.
RHØB(1) = 23.7692E-4
RHØB(2) = 7.0620E-4
RHØB(3) = 7.7650E-5
RHØB(4) = 2.8804E-6
RHØB(5) = 13.9468E-7
RHØB(6) = 4.1189E-8
RHØB(7) = 4.2610E-9
RHØB(8) = 2.2320E-10
RHØB(9) = 1.8450E-12
RHØB(10) = 1.3380E-12
RHØB(11) = 6.1130E-13
PB(1) = 2116.21695
PB(2) = 472.73
PB(3) = 51.979
PB(4) = 2.5155
PB(5) = 1.2181
PB(6) = 2.108E-2
PB(7) = 21.809E-4
PB(8) = 15.562E-5
PB(9) = 75.578E-7
PB(10) = 58.954E-7
PB(11) = 29.759E-7
TB(1) = 518.688
TB(2) = 389.988
TB(3) = 389.988
TB(4) = 508.788
TB(5) = 508.188
TB(6) = 298.188
TB(7) = 298.188
TB(8) = 406.188
TB(9) = 2386.188
TB(10) = 2566.188
TB(11) = 2836.188

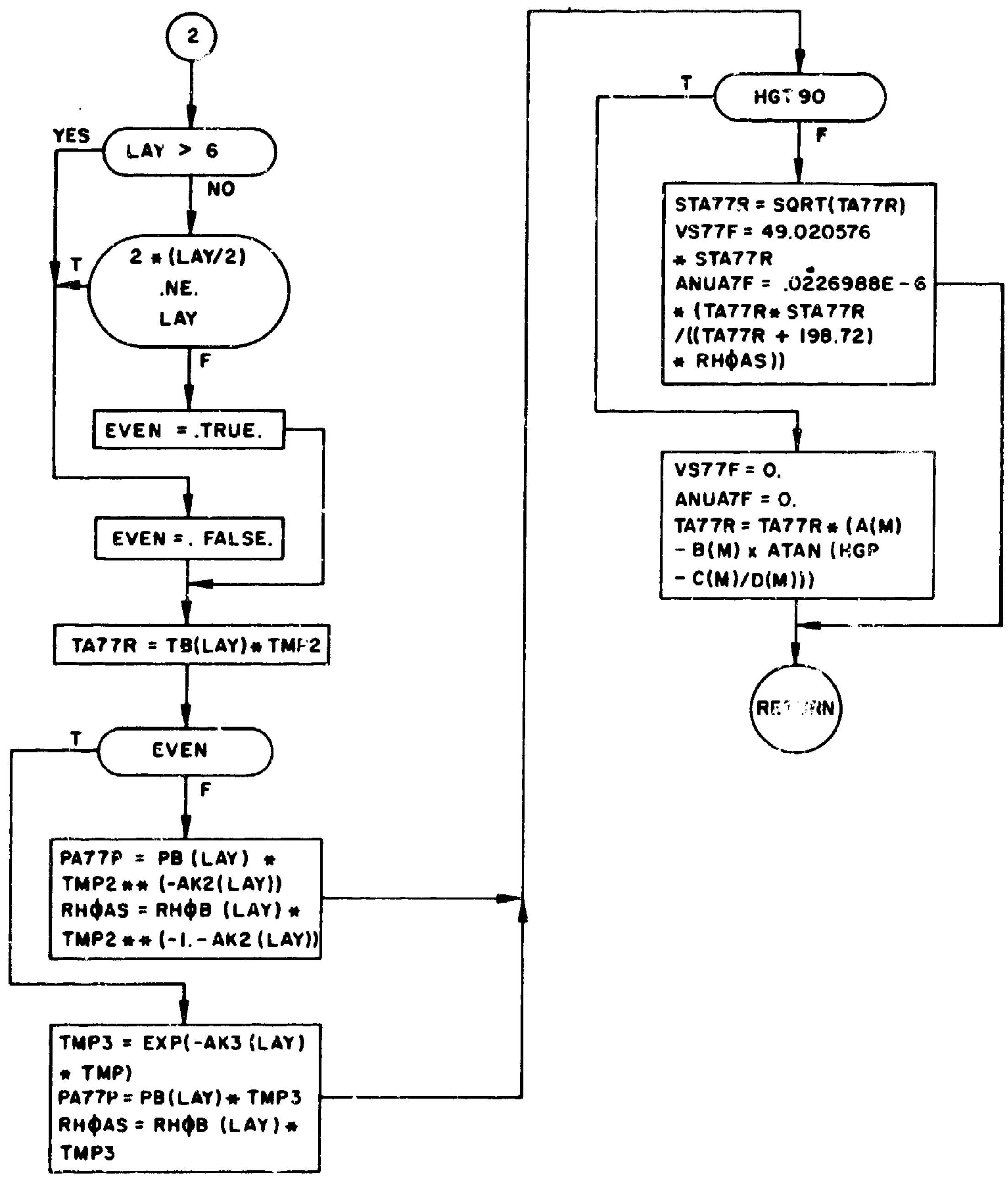
AK1(1) = -.225569E-4
AK1(2) = .0
AK1(3) = .138466E-4
AK1(4) = .0
AK1(5) = -.159202E-4
AK1(6) = .0
AK1(7) = .241458E-4
AK1(8) = .886289E-4
AK1(9) = .754341E-5
AK1(10) = .350715E-5
AK1(11) = .222129E-5
AK2(1) = -5.25612
AK2(2) = 0.
AK2(3) = 11.3883
AK2(4) = 0.
AK2(5) = -7.59218
AK2(6) = 0.
AK2(7) = 8.5412
AK2(8) = 1.70824
AK2(9) = 3.41648
AK2(10) = 6.83296
AK2(11) = 9.76137
AK3(1) = .0
AK3(2) = .157689E-3
AK3(3) = .0
AK3(4) = .120869E-3
AK3(5) = .0
AK3(6) = .206234E-3
AK3(7) = .0
AK3(8) = .0
AK3(9) = .0
AK3(10) = .0
AK3(11) = .0
A(1) .759511
A(2) = .935787
B(1) = .174164
B(2) = .273966
C(1) = 220.
C(2) = 180.
D(1) = 25.
D(2) = 140.

1

FLOW DIAGRAM (ATMS)



FLOW DIAGRAM (ATMS)



c. Usage - Linkage is obtained via the statement:

CALL INVR3 (A,B, INDER)

where

A = the array name of the matrix to be inverted.

B = the array name where the resulting matrix is to be stored.

INDER = an error indicator set by the routine.

(a) INDER = 1, results are good

(b) INDER = 2, A was 0.

18. MULT31 - A MATRIX MULTIPLICATION ROUTINE

a. Purpose - To postmultiply a 3 x 3 matrix by a 3 x 1 matrix.

b. Method - The result of [A] [B] = [C] is computed using single precision floating point arithmetic. All elements must be stored in the normal FORTRAN sense (i.e., columnwise).

c. Usage - The matrix multiplication is obtained by the statement:

CALL MULT31(A,B,C)

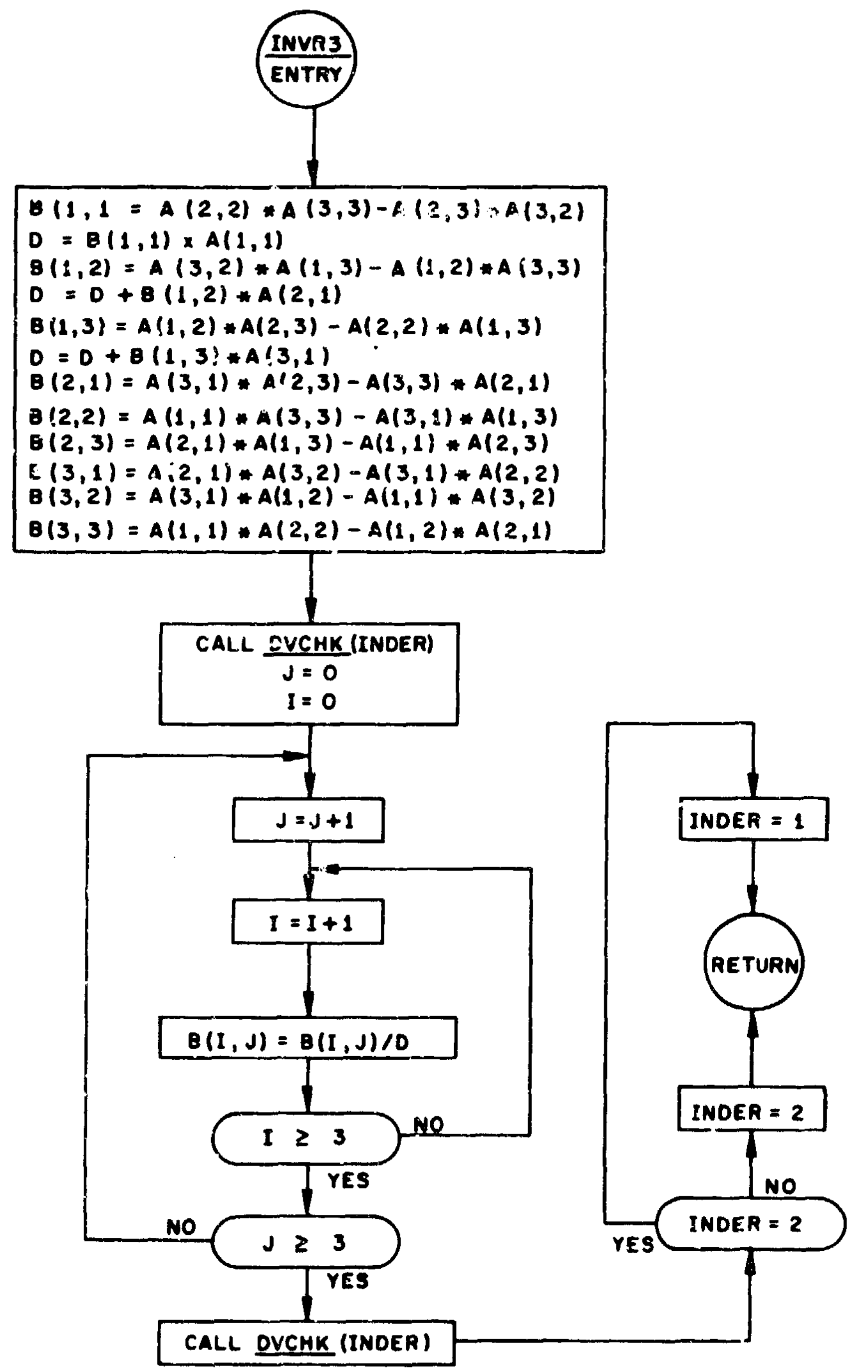
where

A = array name of the 3 x 3 matrix [A]

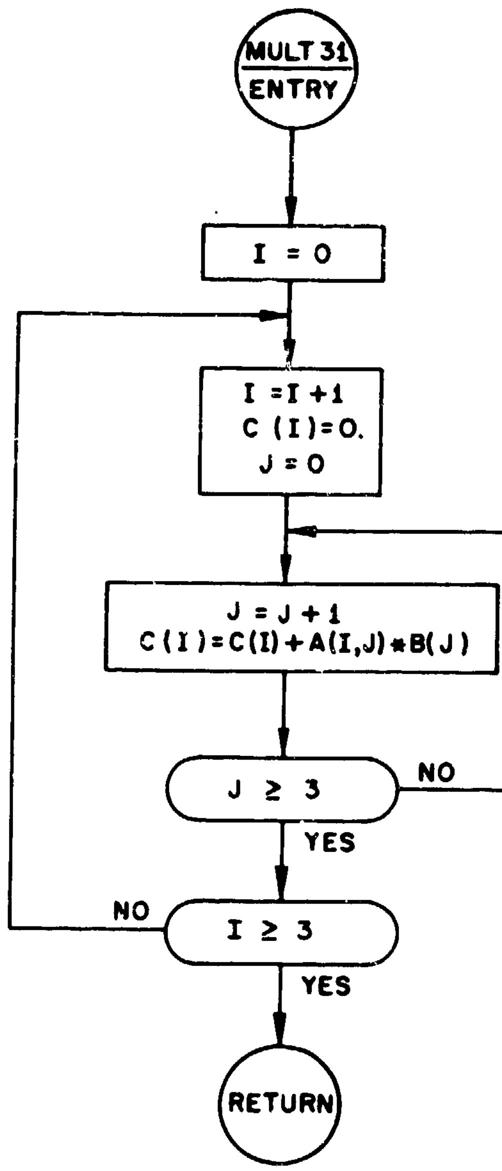
B = array name of the 3 x 1 matrix [B]

C = array name of the resulting 3 x 1 matrix [C]

FLOW DIAGRAM (INVR3)



FLOW DIAGRAM (MULT 31)



19. TRNPØS - A 3 x 3 MATRIX TRANSPOSE ROUTINE

a. Purpose - To transpose a 3 x 3 matrix [A] to obtain the 3 x 3 matrix [A]'.

b. Method - The resulting transposed matrix is stored in a separate array. All elements of [A] must be stored in the normal FORTRAN sense (i.e., columnwise).

c. Usage - The transpose of a 3 x 3 matrix [A] is obtained by:

CALL TRNPØS (A,B)

where

A = The array name of the 3 x 3 matrix [A].

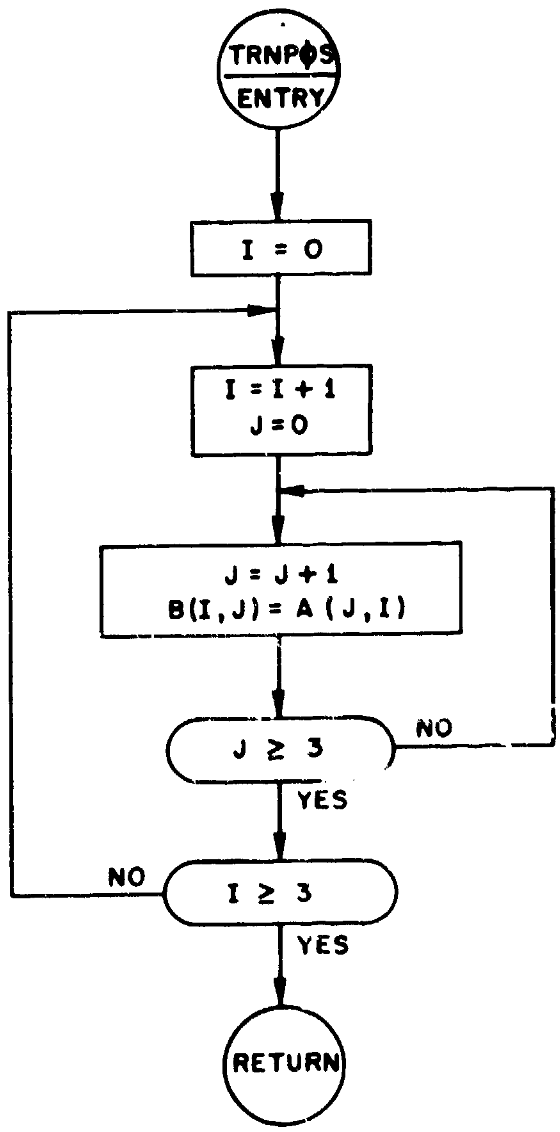
B = The array name of the 3 x 3 matrix [A].

20. HIHØ - N-DIMENSIONAL TABLE CALL ROUTINE

a. Purpose - To set up the NA array and Z location of tables with dimension from 3 to 5 as required by the calling sequence to NDTLU, which is

CALL NDTLU(ND,NA,X,A,XA,ZR,IE), to make the call to NDTLU, and return the function value or data on a table read error.

FLOW DIAGRAM (TRNPØS)



b. Usage - Linkage to the subroutine is made via the statement

CALL HIHØ (N, LØCT, NX1, NX2, NX3, NX4, X1ARG, X2ARG, X3ARG, X4ARG, A)

where

N = dimension of table look-up, when A = F(x), N=2.

LØCT = location of the first value in the table.

NX1 to NX4 = location of number of points in the X1 to X4 array of independent variable values

X1ARG to X4ARG = name of X1 to X4 argument or a dummy location if N < 5.

A = location of the dependent variable.

21. TLU - TWO-DIMENSIONAL TABLE LOOK-UP ROUTINE

a. Purpose - Given an argument X, to compute Y = F(X) from a table of X and Y values by linear interpolation.

b. Method - The table of X values is searched until for some

i, $X_i < X < X_{i+1}$

Linear interpolation is then performed. If, for some i, $X=X_i$ then Y is set to Y_i.

c. Usage - Entry is made via the statement,

CALL TLU (X, LØCT, Y)

where

X = = variable name of the argument

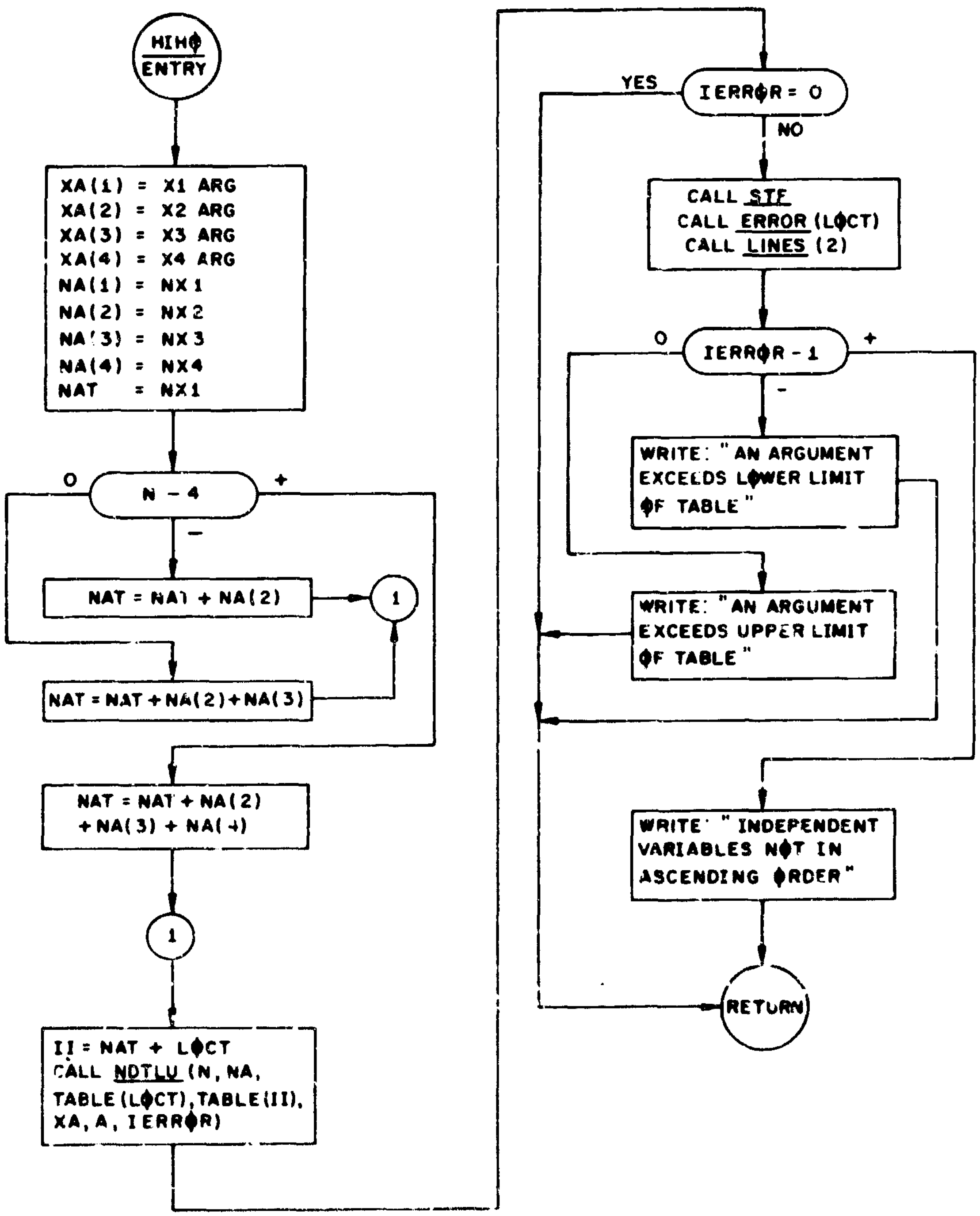
LØCT = location of first subscript of desired table of data

Y = variable name of the interpolated value.

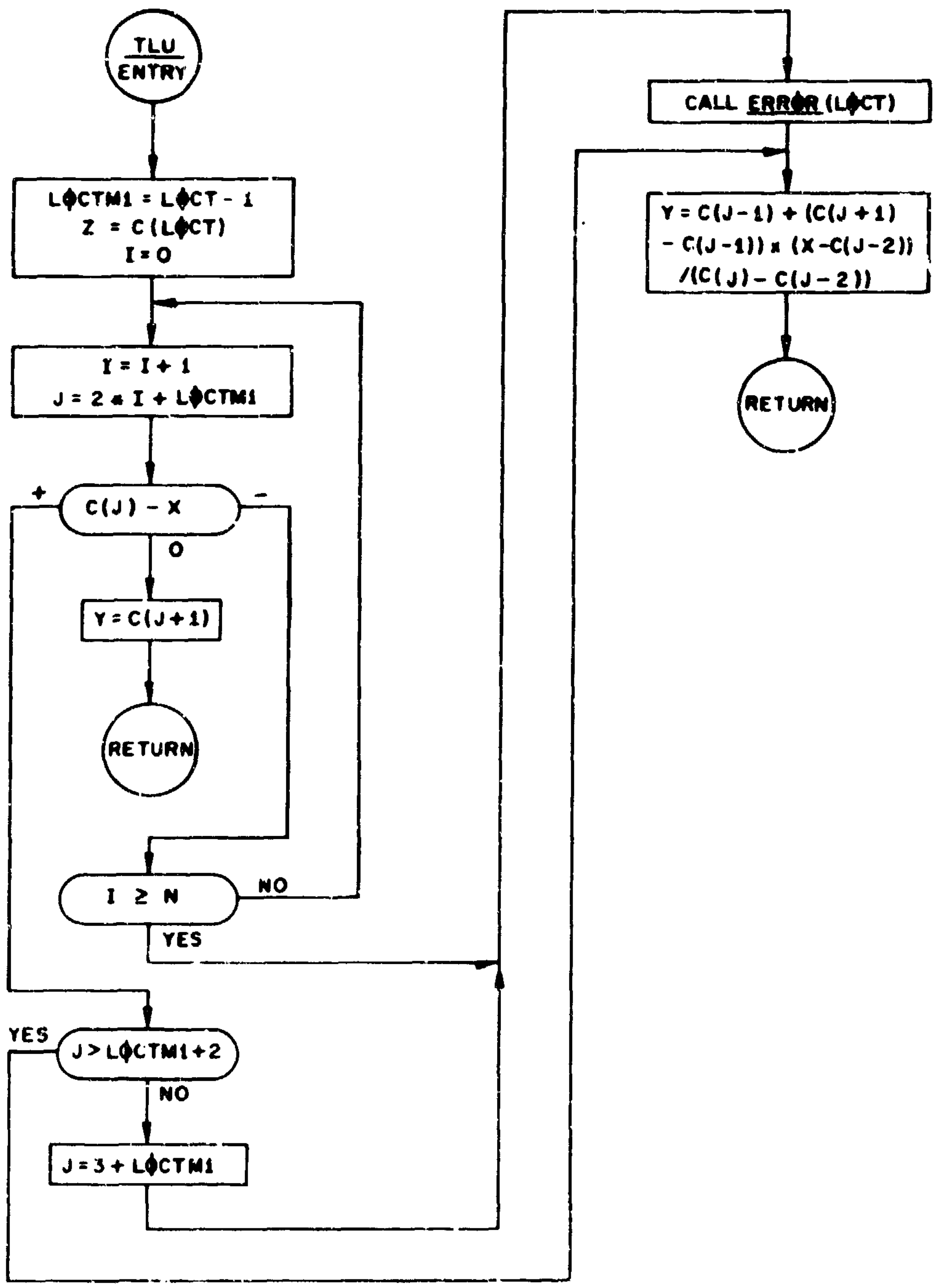
All tables are stored in a table called C. Therefore, the first value of a particular table is located at C(LØCT). As an example, assume that the name of the desired table is FTAB01. Then the curve must be stored as follows:

C(LØCT) = FTAB01 (1) = N=No. of points in curve (integer)

FLOW DIAGRAM (HIHØ)



FLOW DIAGRAM (TLU.)



C(LØCT+1) = FTAB01 (2) = X_1

C(LØCT+2) = FTAB01 (3) = Y_1

C(LØCT+3) = FTAB01 (4) = X_2

C(LØCT+4) = FTAB01 (5) = Y_2

. .

. .

. .

. .

C(LØCT+2N) = FTAB01 (2N) = X_N

C(LØCT+2N+1)= FTAB01 (2N+1) = Y_N

22. TFFS1 - ENGINE THRUST AND THROTTLE SETTING

a. Purpose - To provide a method of introducing the engine thrust characteristics into the computation, with an option to find the throttle setting that corresponds with a certain thrust and Mach number.

b. Usage - Linkage to TFFS is accomplished via the following statements:

(1) CALL TFFS1

Pre-data initialization.

(2) CALL TFFS3

Thrust computation section.

(3) CALL TFFS4

Initial print

(4) CALL TFFS5

Code printing to identify the coming time history.

(5) CALL TFFS6

Time History Print

(6) CALL TFFS7

Update integration (none for this subprogram)

23. VPCS - Vehicle Physical Characteristics

a. Purpose - To introduce various physical characteristics into the general solution of the problem. Here, mass, moments, and products of inertia of the vehicle and rotating machinery, reference lengths and areas for aerodynamic coefficients, jet damping characteristics lengths, and center of gravity information are included.

b. Usage - Linkage to VPCS is provided by the following statements:

(1) CALL VPCS1

Pre-data initialization. Necessary nominal values are set.

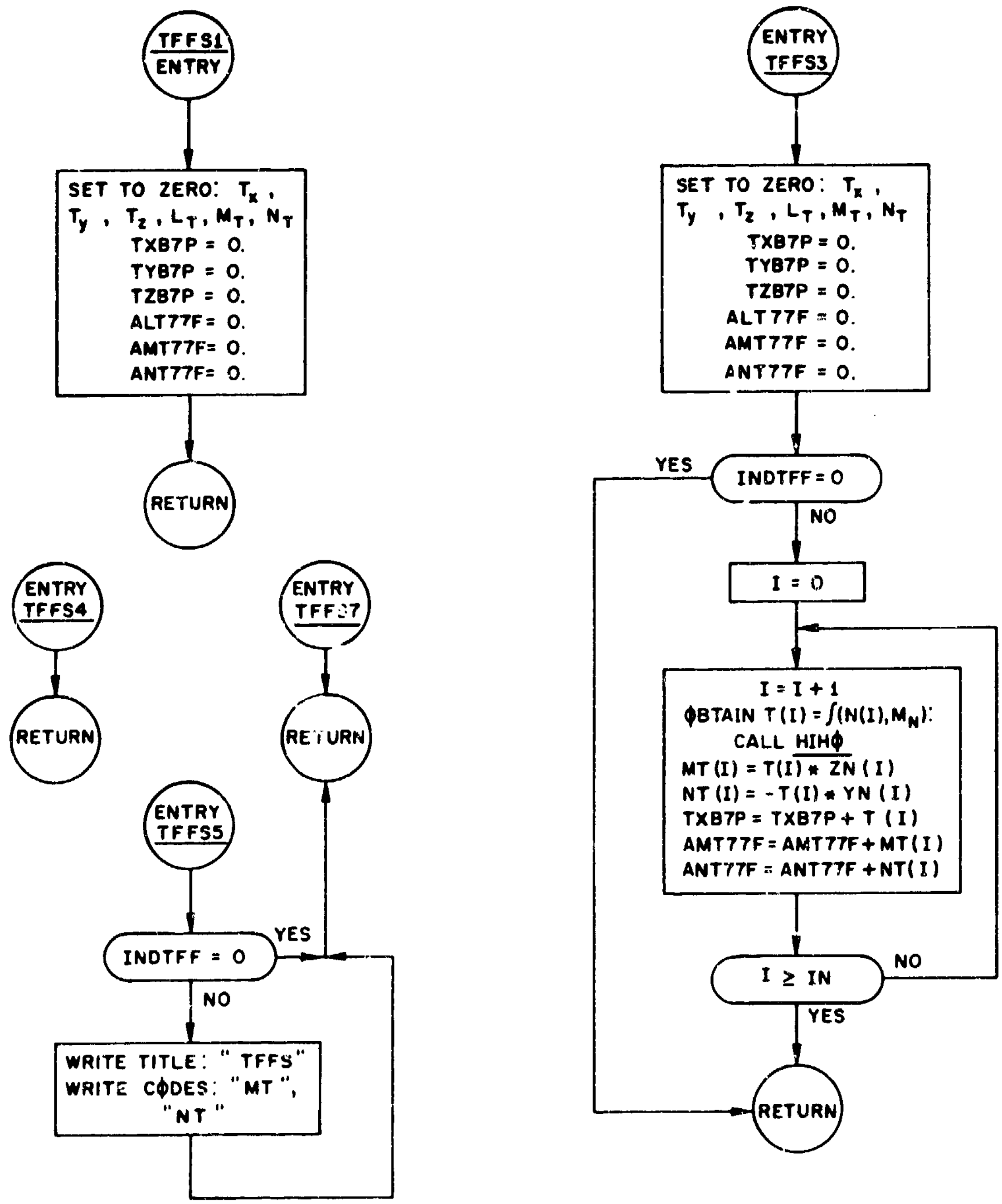
(2) CALL VPCS2

Post-data initialization.

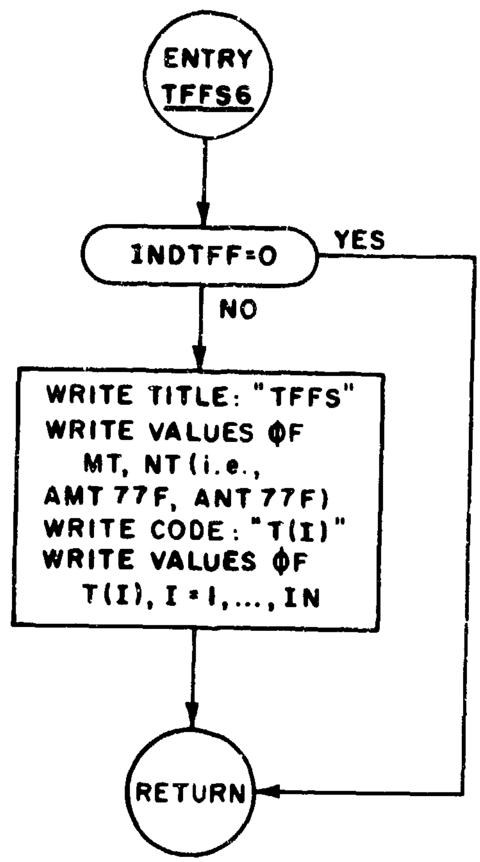
(3) CALL VPCS3

VPCS computations are performed if INDVPC is non-zero.

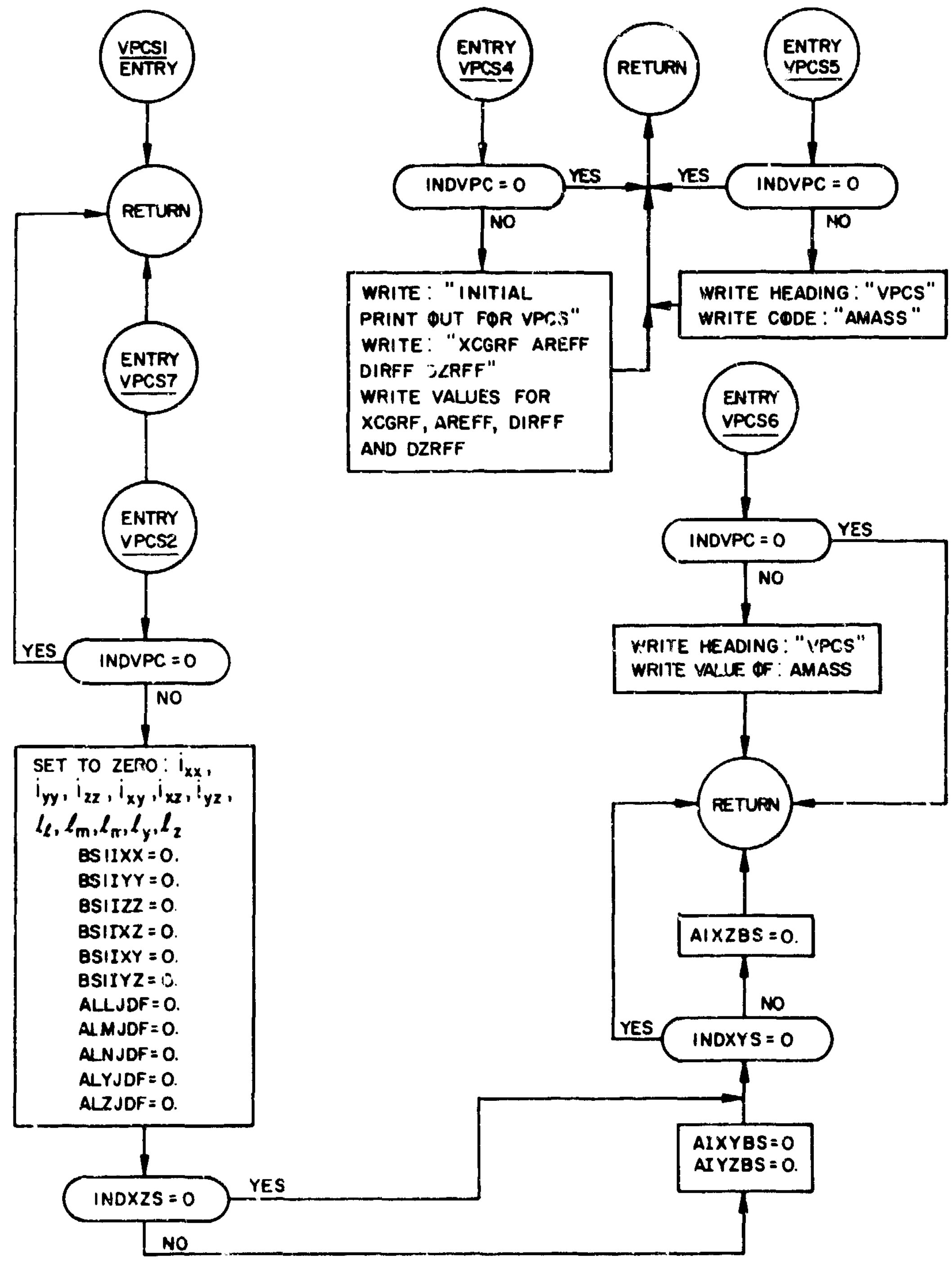
FLOW DIAGRAM (TFFS1)



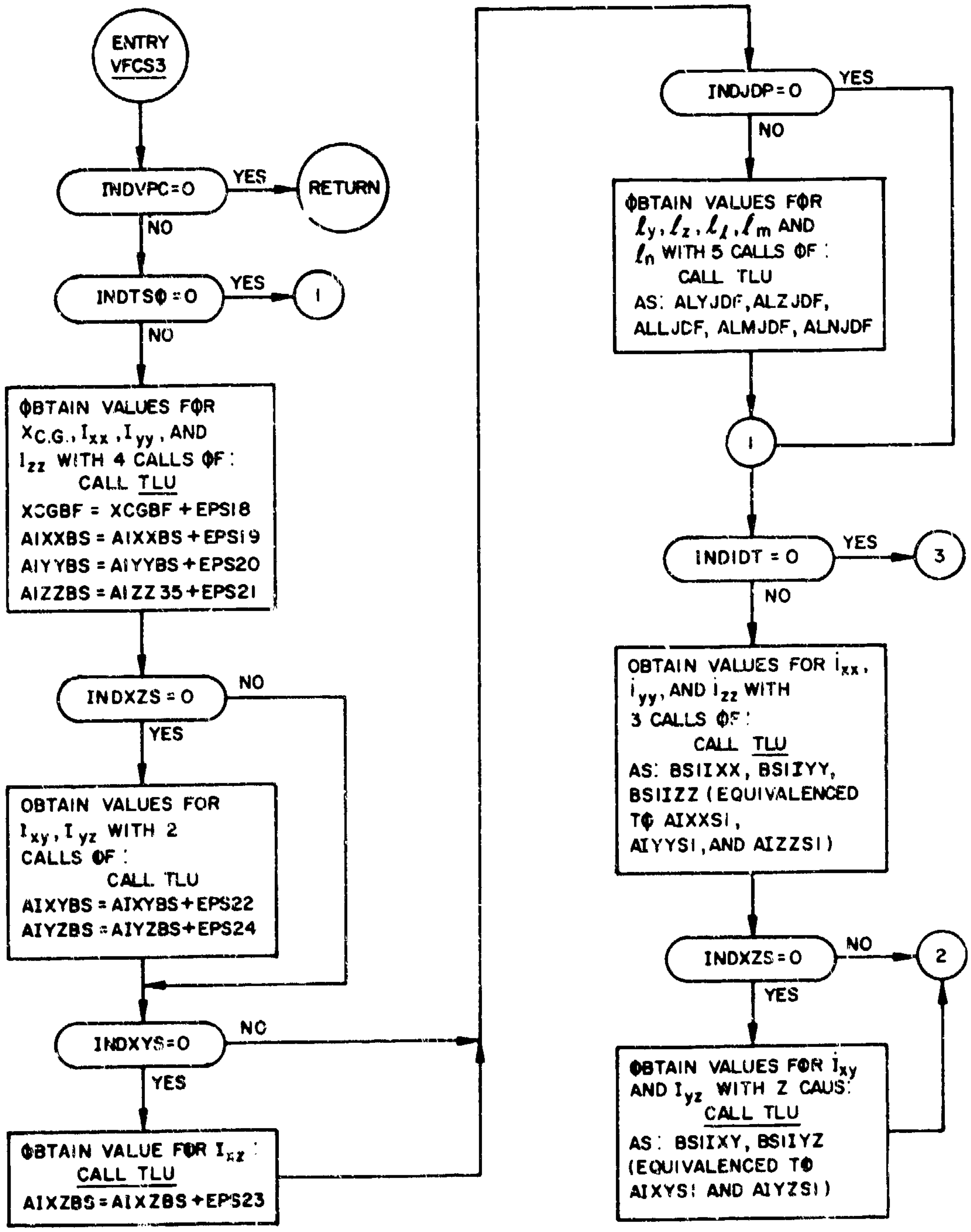
FLOW DIAGRAM (TFFS!)



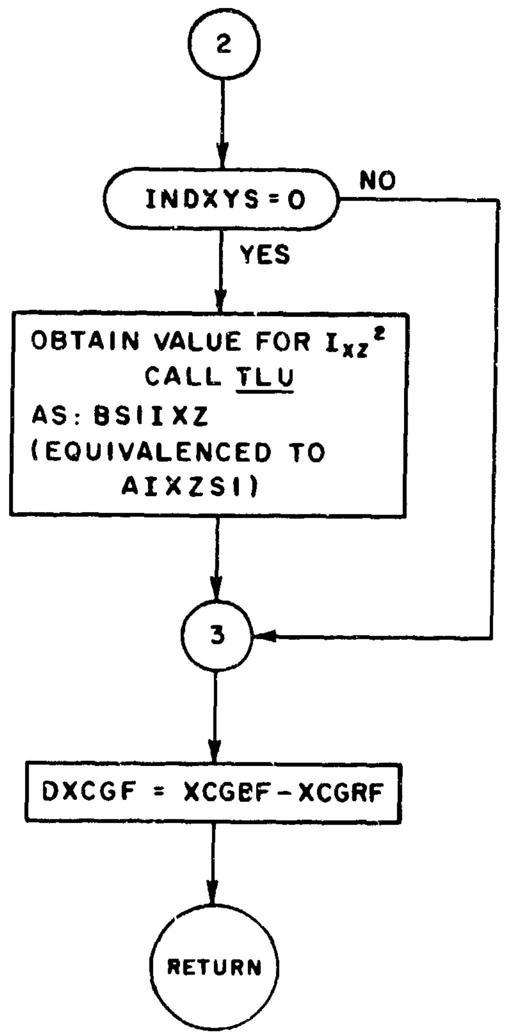
FLOW DIAGRAM (VPCSI)



FLOW DIAGRAM (VPCSI)



FLOW DIAGRAM (VPCSI)



(4) CALL VPCS4

Initial print

(5) CALL VPCS5

Code printing to identify the coming time history is performed if INDVPC $\neq$ 0.

(6) CALL VPCS6

Time history print.

24. SACS - AERODYNAMIC FORCES AND MOMENTS

a. Purpose - To provide a complete accounting of the various contributions to the aerodynamic forces and moments, regardless of the flight conditions or the vehicle considered. In SACS coefficients are computed in the proper coordinate system for use in other parts of the TOLA program.

b. Usage - Linkage to SACS is accomplished via the following statements:

(1) CALL SACS1

Pre-data initialization.

(2) CALL SACS3

Aerodynamic computation.

(3) CALL SACS4

Initial print. (None for this subprogram).

(4) CALL SACS5

Code print to identify the trajectory point for SACS.

(5) CALL SACS6

Time history print for SACS.

(6) CALL SACS7

Update integration (None for this subprogram).

(7) CALL SACS8 (C_{LR}, α , C_{DR})

Determine the α_d that corresponds with C_L and C_D.

(8) CALL SACS9 (α , C_L, C_D)

Determine the C_L and C_D that corresponds with α .

(9) CALL SACS10 (α)

Determine δ_{qn} as a function of α_d and all current variables.

(10) CALL SACS11 (δ_{rN})

Determine δ_{r_N} as a function of all current variables.

25. AERØ1 - AERODYNAMIC DATA LOOKUP FUNCTION

a. Purpose - To look up aerodynamic data from the table array in CØMMON TBDIR/C(300). It looks up the first two values of a particular table depending on the argument of the function.

b. Usage - Linkage to AERØ1 is accomplished in the following ways:

Y = AERØ1 (LØCT, AERØ2)

where LØCT = subscript for the table C(300) which will be the 1st location of a particular table that one is interested in.

After the function is executed,

AERØ1 = 1st value of the table that begins at C(LØCT)

AERØ2 = 2nd value of the table that is located at C(LØCT + 1)

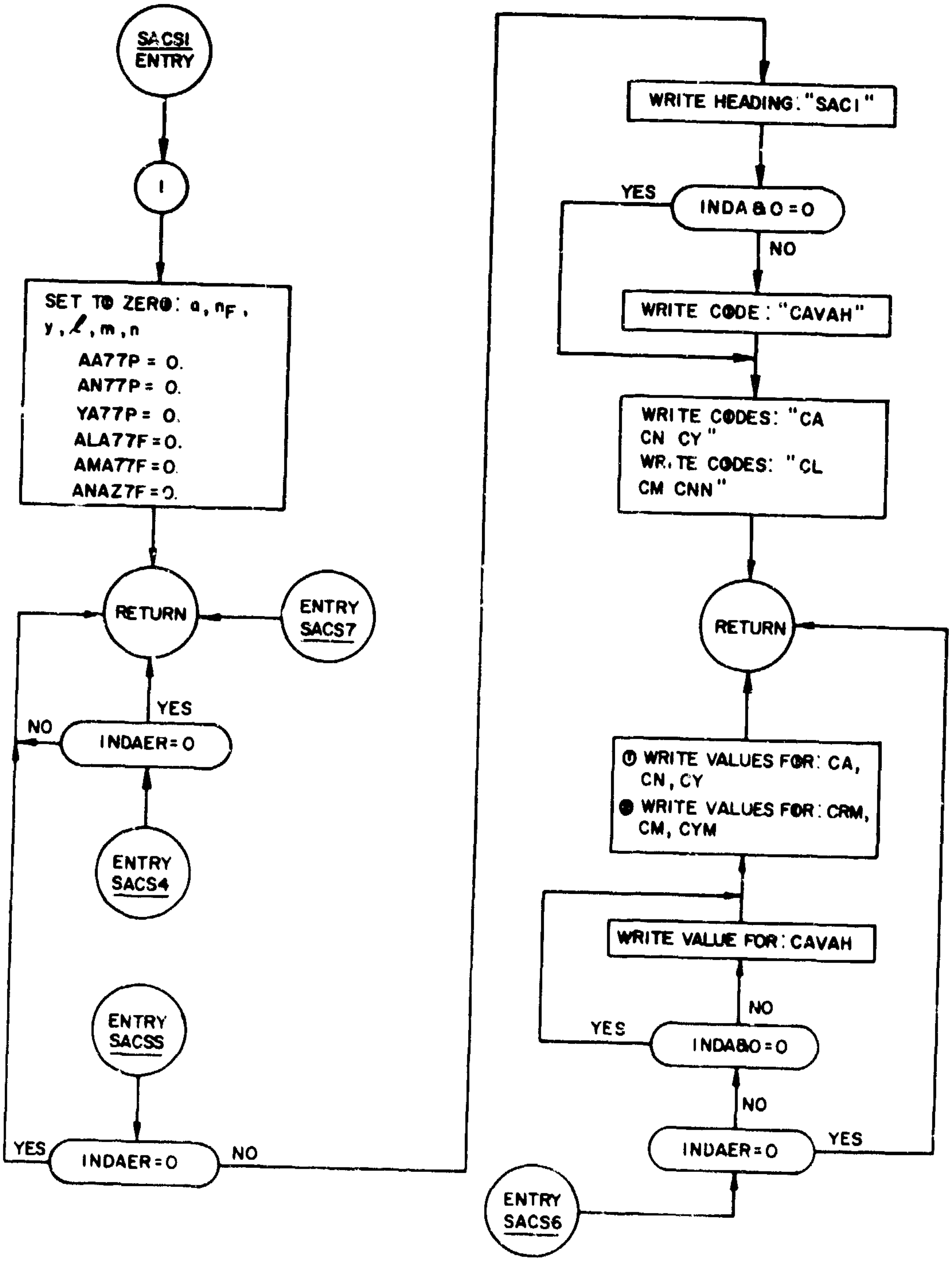
26. AQUAD - A QUADRATIC FUNCTION

a. Purpose - To solve the equation $a|x|x + b x + C = 0$

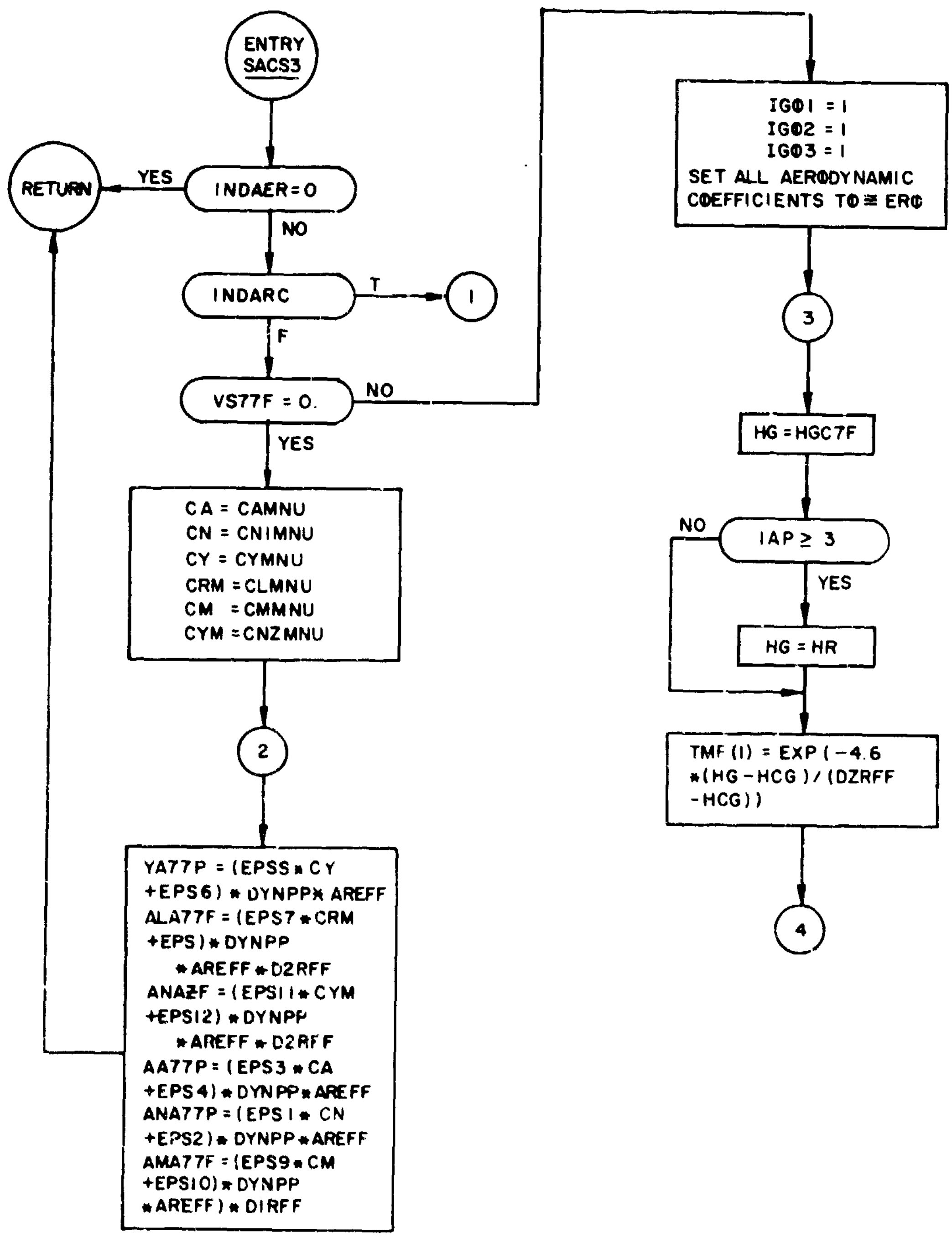
b. Usage - Linkage to AQUAD is accomplished in the following way:

Y = AQUAD (A, B, C, XLLIM, XULIM)

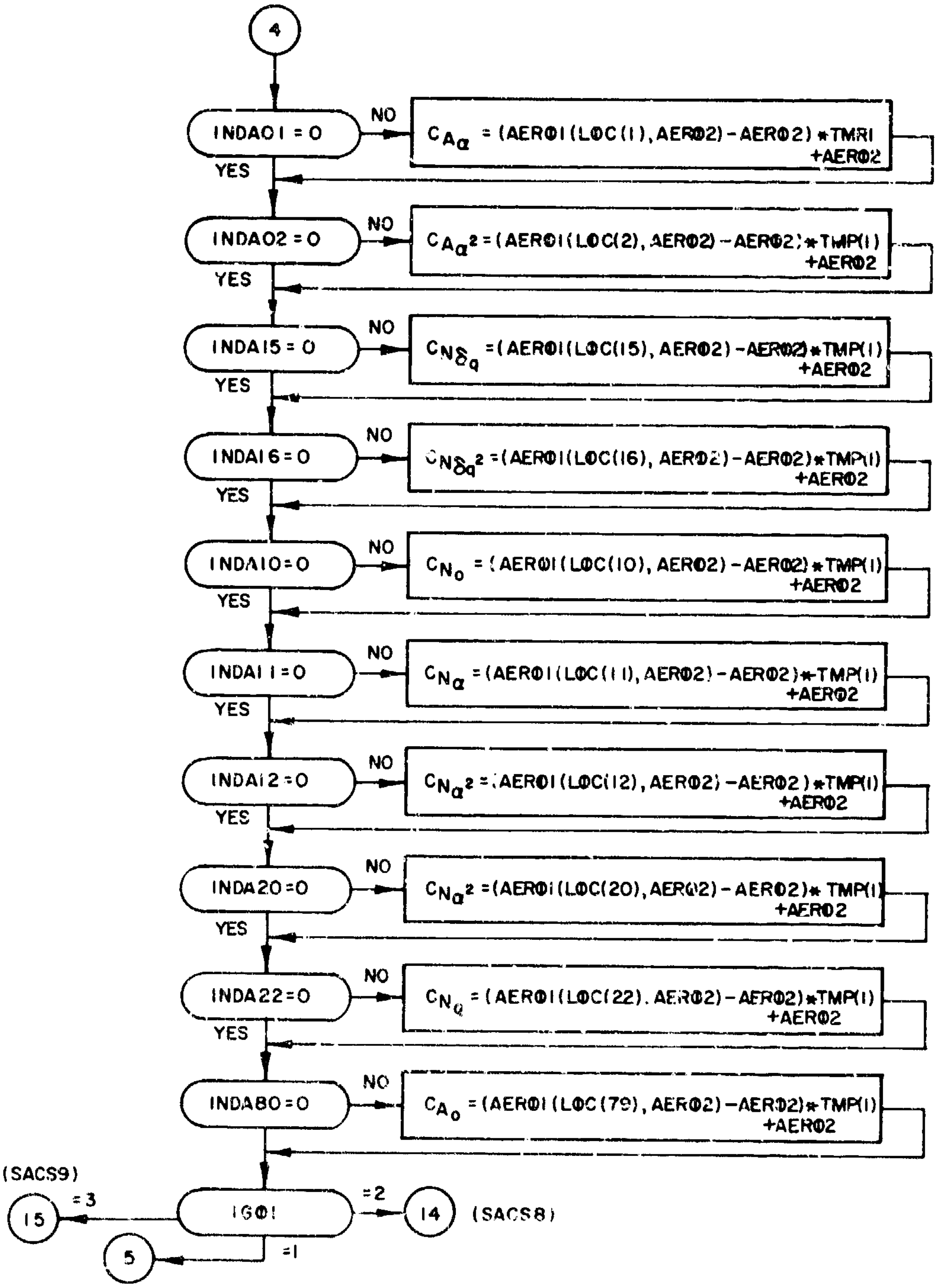
FLOW DIAGRAM (SACS1)



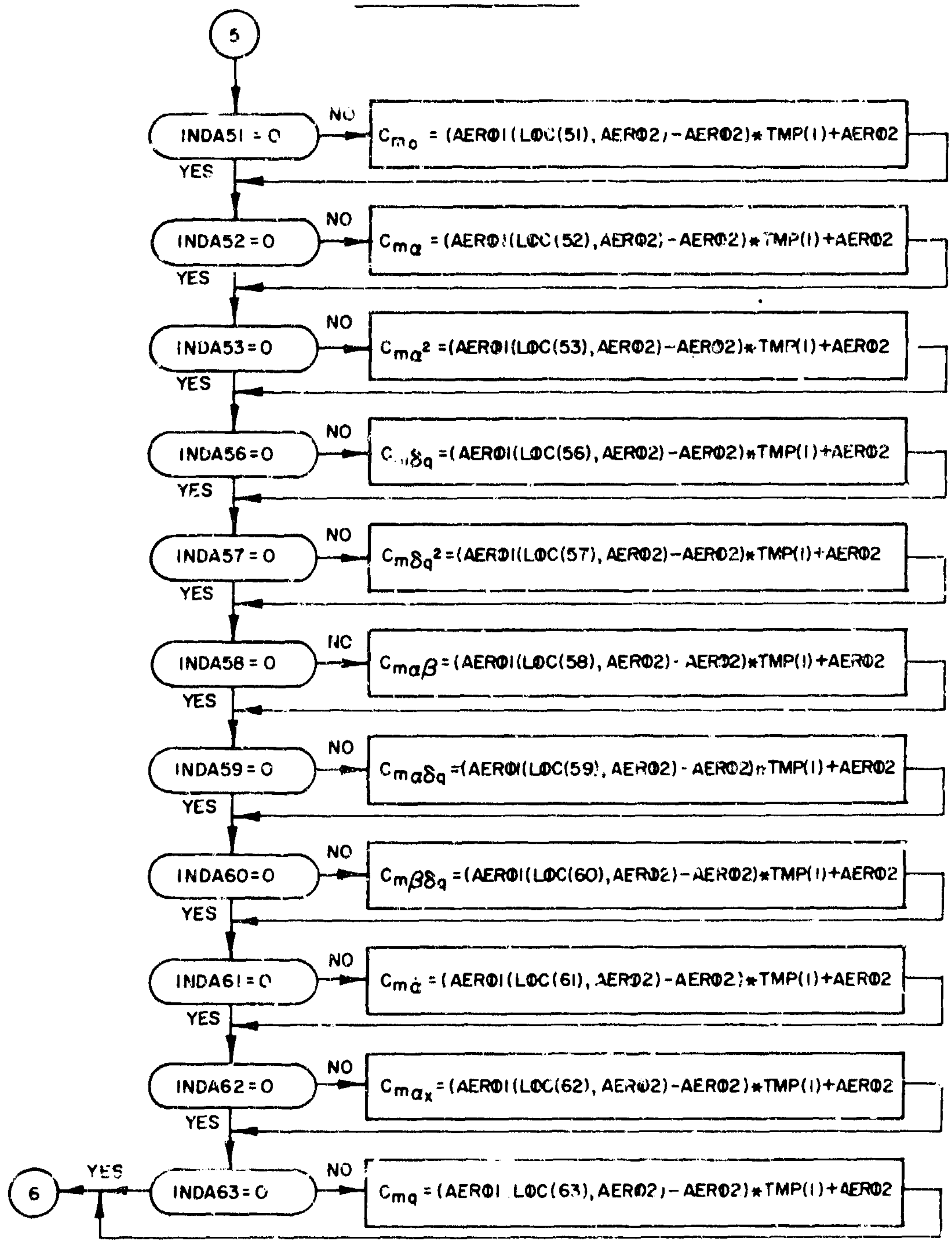
FLOW DIAGRAM (SACSI)



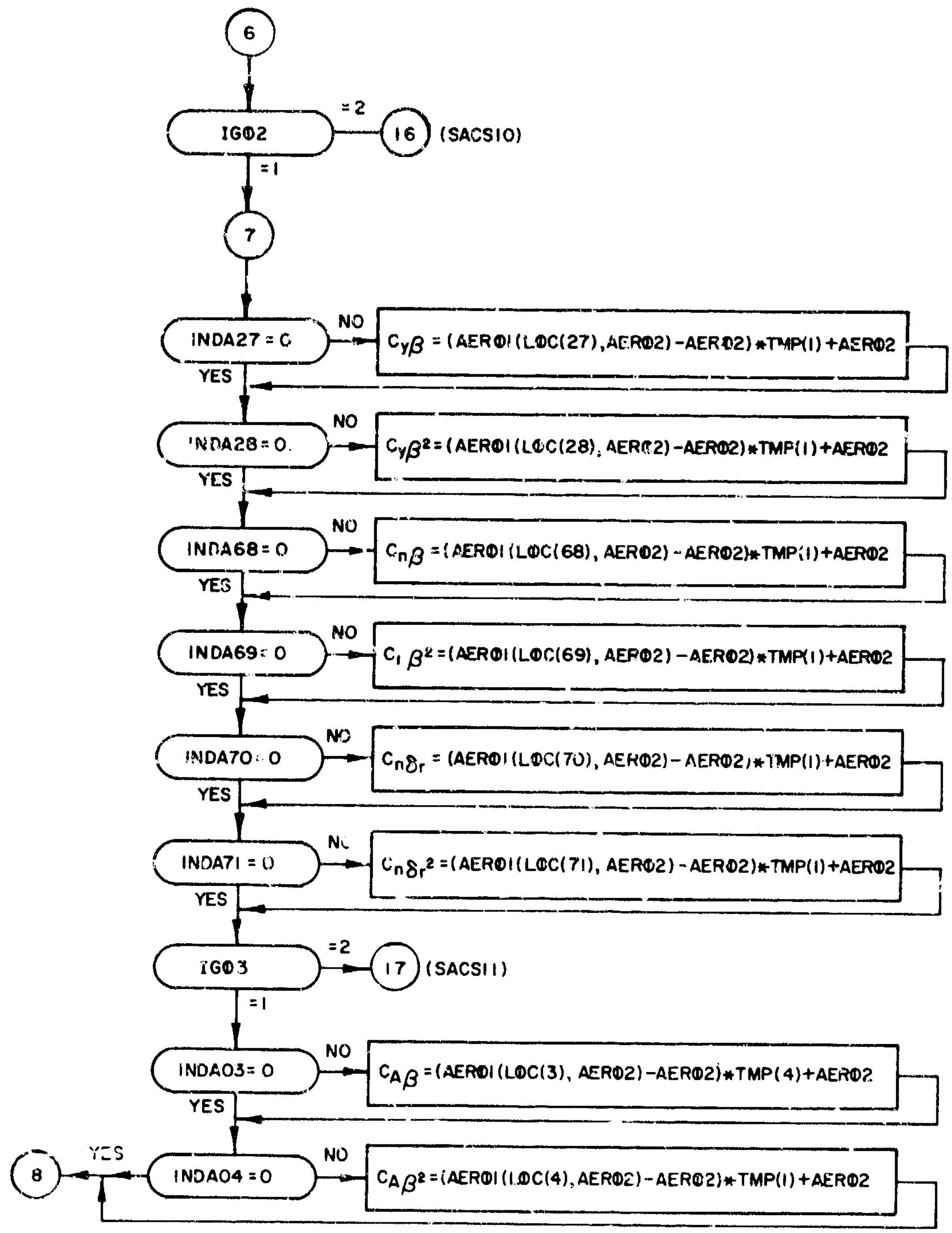
FLOW DIAGRAM (SACS!)



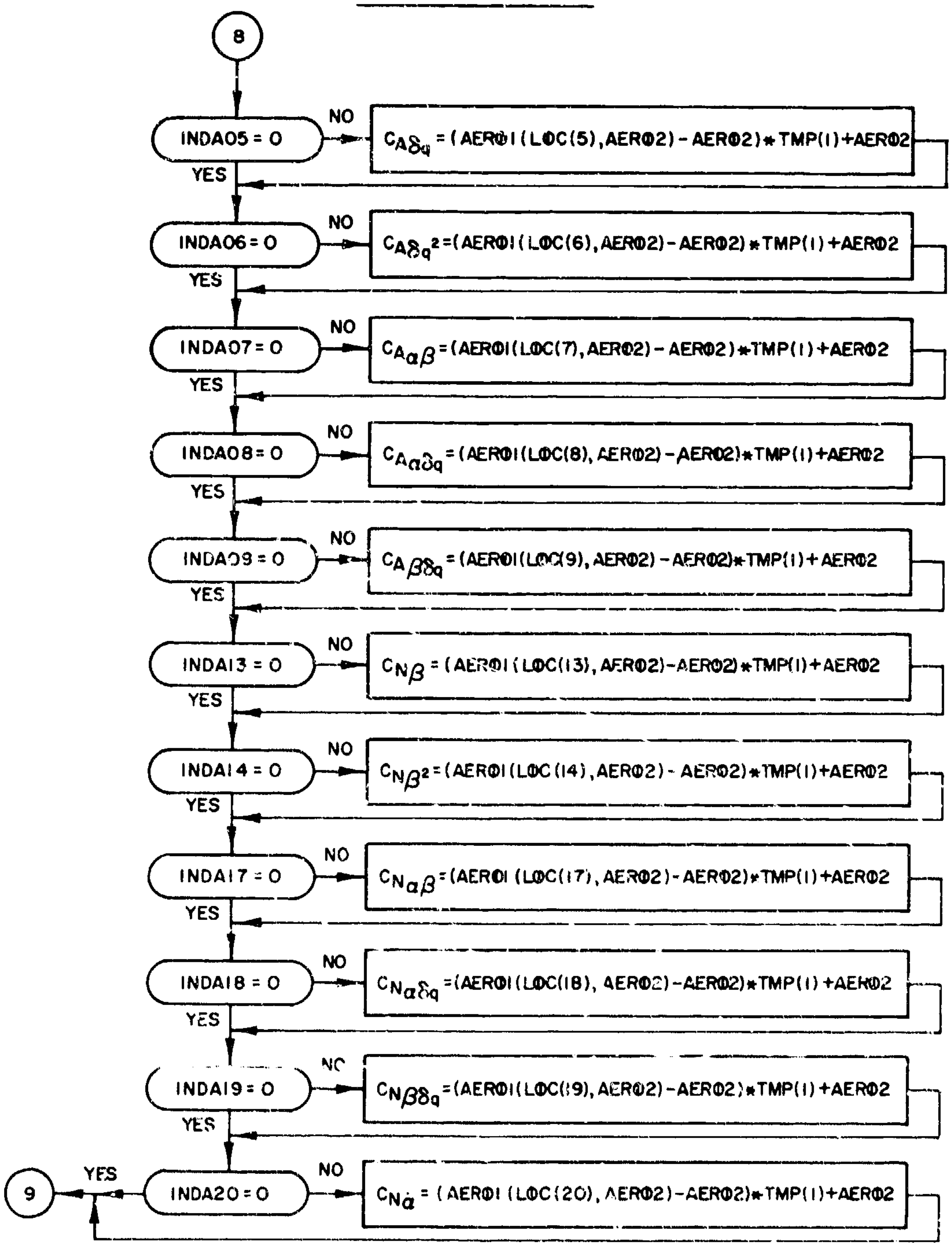
FLOW DIAGRAM (SACSI)



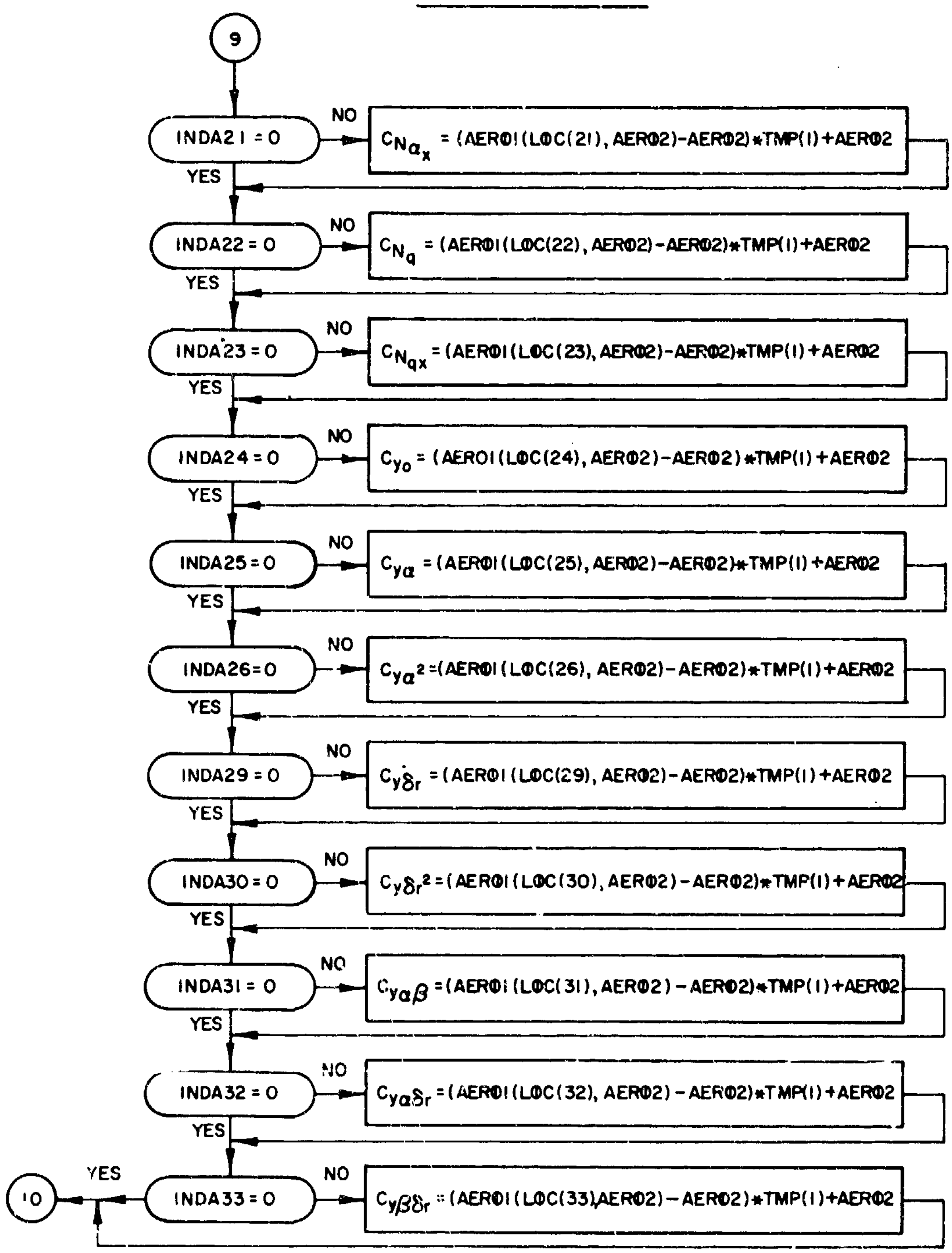
FLOW DIAGRAM (SACS1)



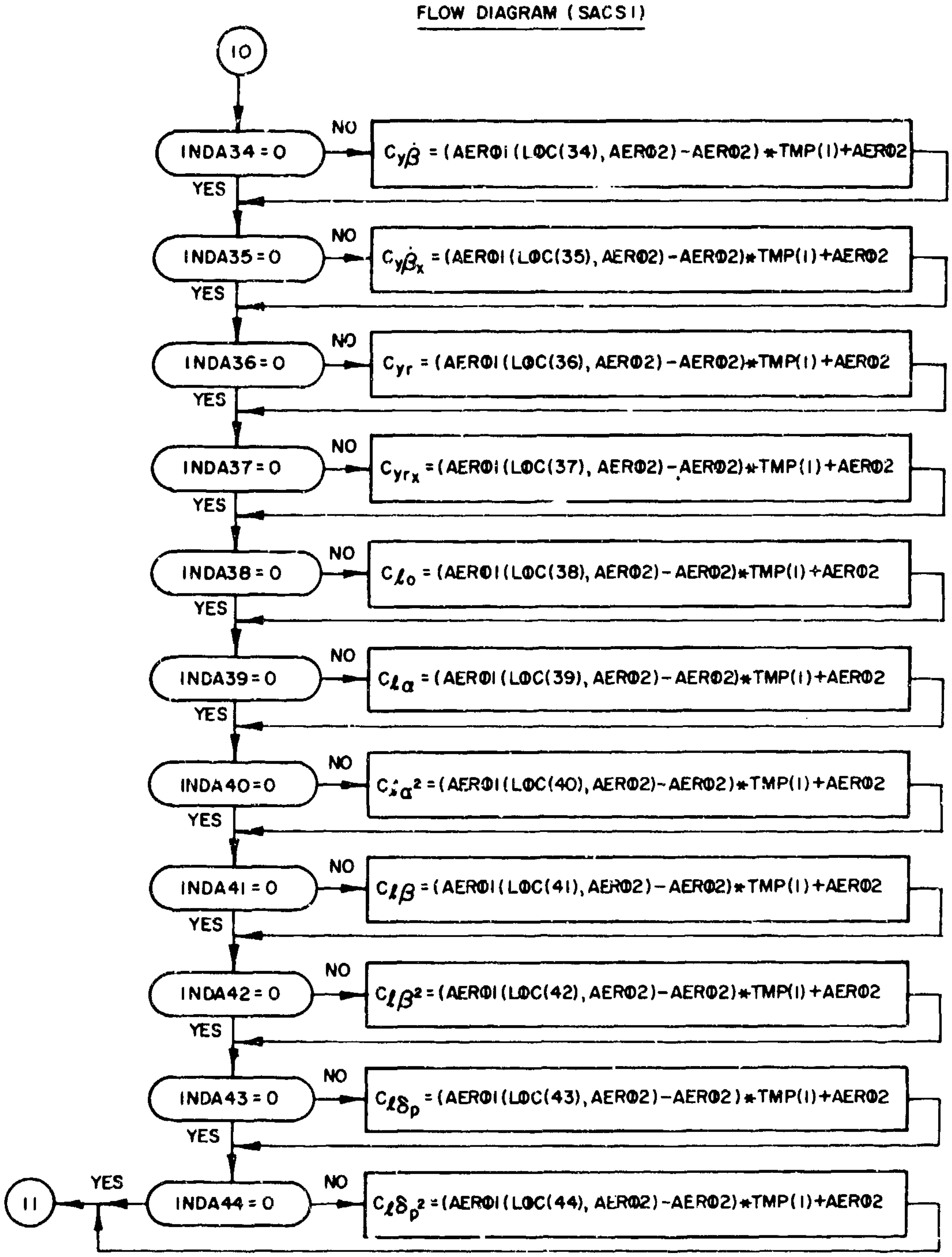
FLOW DIAGRAM (SACS1)



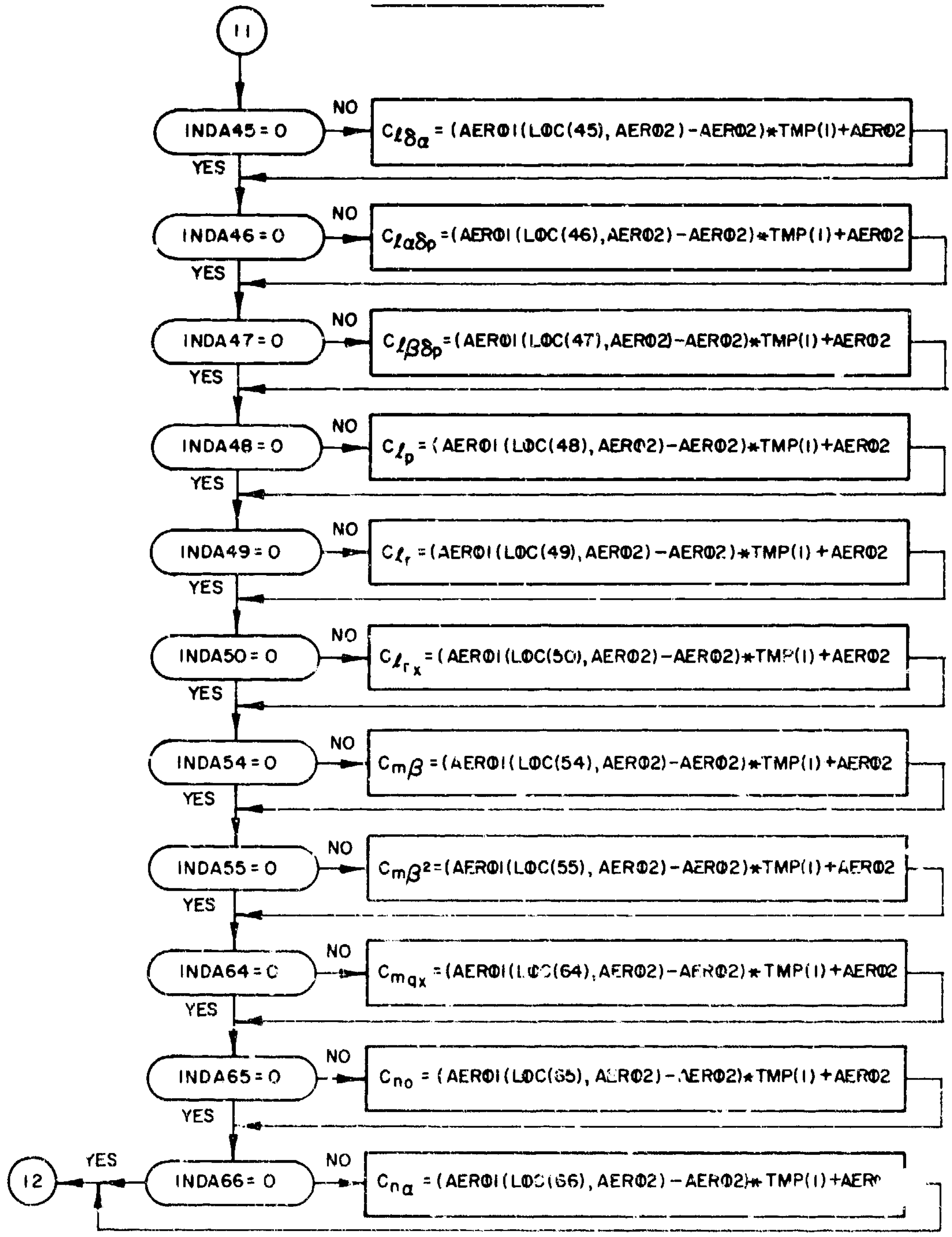
FLOW DIAGRAM (SACS1)



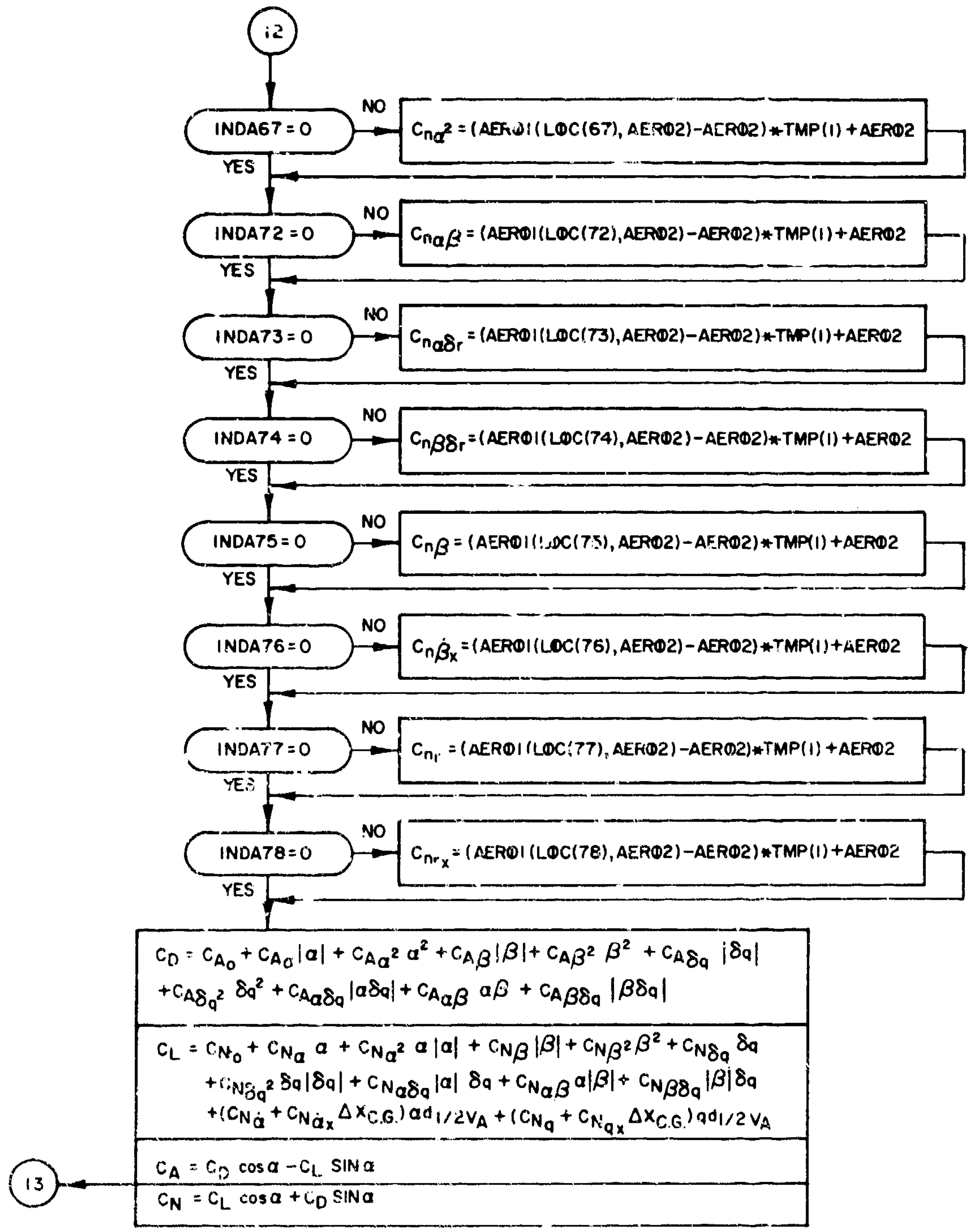
FLOW DIAGRAM (SACS1)



FLOW DIAGRAM (SACSI)



FLOW DIAGRAM (SACSI)



FLOW DIAGRAM (SACS I)

13

$$
\begin{aligned}
C_y = {} & C_{y_0} + C_{y_\alpha}|\alpha| + C_{y_{\alpha^2}}\alpha^2 + C_{y_\beta}\beta + C_{y_{\beta^2}}\beta|\beta| \\
& + C_{y_{\delta_r}}\delta_r + C_{y_{\delta_r^2}}|\delta_r|\delta_r + C_{y_{\alpha\delta_r}}|\alpha|\delta_r + C_{y_{\alpha\beta}}|\alpha|\beta \\
& + C_{y_{\beta\delta_r}}|\beta|\delta_r + (C_{y_{\dot\beta}} + C_{y_{\dot\beta_x}}\Delta X_{C.G.})\dot\beta d_2/2V_A \\
& + (C_{y_r} + C_{y_{r_x}}\Delta X_{C.G.})\, r d_2/2V_A
\end{aligned}
$$

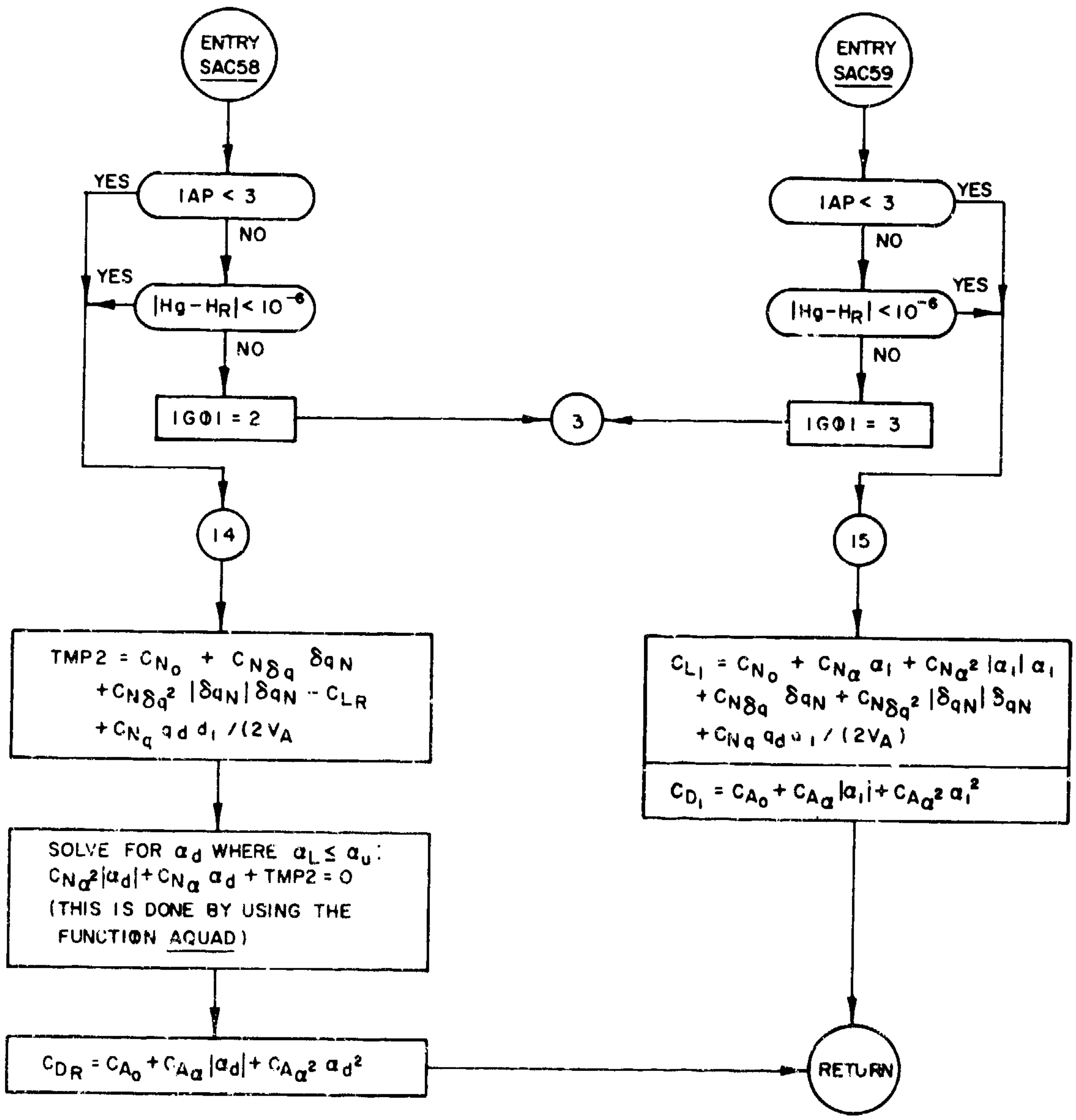
$$
\begin{aligned}
C_{RM} = {} & C_{\ell_0} + C_{\ell_\alpha}|\alpha| + C_{\ell_{\alpha^2}}\alpha^2 + C_{\ell_\beta}|\beta| + C_{\ell_{\delta_p}}\delta_p \\
& + C_{\ell_{\delta_p^2}}|\delta_p|\delta_p + C_{\ell_{\alpha\delta_p}}|\alpha|\delta_p + C_{\ell_{\alpha\beta}}|\alpha\beta| \\
& + C_{\ell_{\beta\delta_p}}|\beta|\delta_p + C_{\ell_p}\, p d_2/2V_A + (C_{\ell_r} + C_{\ell_{r_x}}\Delta X_{C.G.})\, r d_2/2V_A
\end{aligned}
$$

$$
\begin{aligned}
C_m = {} & C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\alpha^2}}\alpha|\alpha| + C_{m_\beta}|\beta| + C_{m_{\beta^2}}\beta^2 \\
& + C_{m_{\delta_q}}\delta_q + C_{m_{\delta_q^2}}\delta_q|\delta_q| + C_{m_{\alpha\delta_q}}|\alpha|\delta_q \\
& + C_{m_{\alpha\beta}}\alpha|\beta| + C_{m_{\beta\delta_q}}|\beta|\delta_q + (C_{m_{\dot\alpha}} + C_{m_{\dot\alpha_x}}\Delta X_{C.G.})\dot\alpha d_1/2V_A \\
& + (C_{m_q} + C_{m_{q_x}}\Delta X_{C.G.})\, q d_1/2V_A - C_N\Delta X_{C.G.}/d_1
\end{aligned}
$$

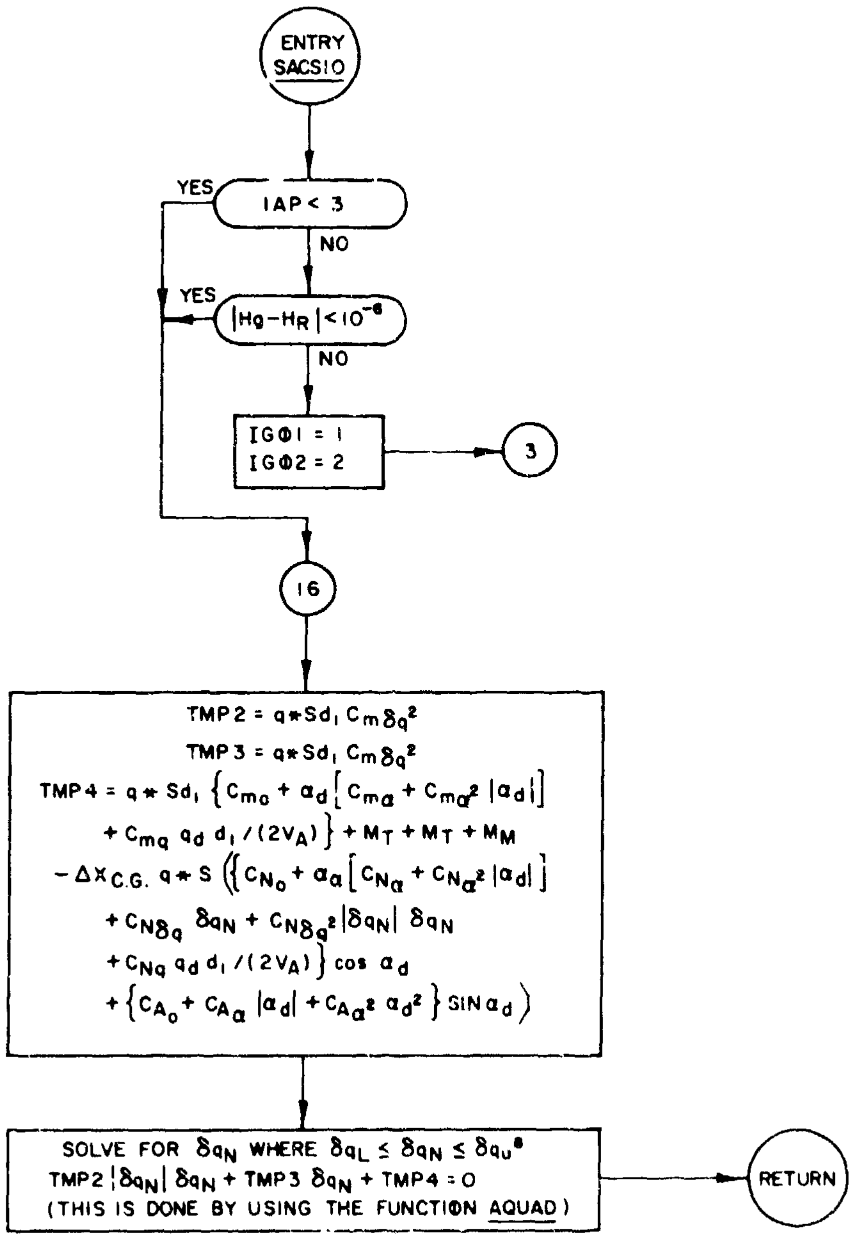
$$
\begin{aligned}
C_{YM} = {} & C_{n_0} + C_{n_\alpha}|\alpha| + C_{N_{\alpha^2}}\alpha^2 + C_{n_\beta}\beta + C_{n_\beta}\beta + C_{n_{\beta^2}}\beta|\beta| \\
& + C_{n_{\delta_r}}\delta_r + C_{n_{\delta_r^2}}|\delta_r|\delta_r + C_{n_{\alpha\delta_r}}|\alpha|\delta_r + C_{n_{\alpha\beta}}|\alpha|\beta \\
& + C_{n_{\beta\delta_r}}|\beta|\delta_r + (C_{n_{\dot\beta}} + C_{n_{\dot\beta_x}}\Delta X_{C.G.})\dot\beta d_2/2V_A \\
& + (C_{n_r} + C_{n_{r_x}}\Delta X_{C.G.})\, r d_2/2V_A - C_y\Delta X_{C.G.}/d_2
\end{aligned}
$$

2

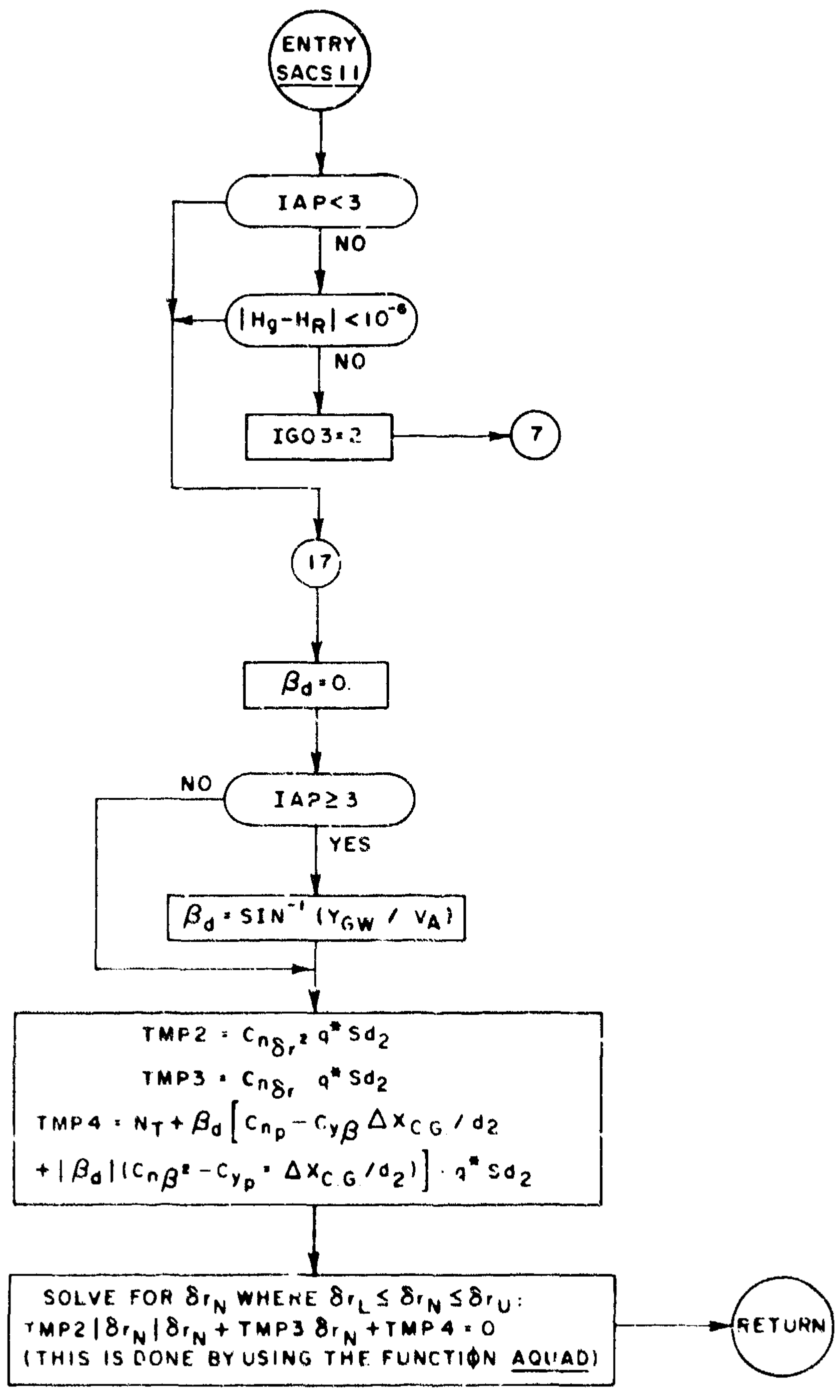
FLOW DIAGRAM (SACS1)



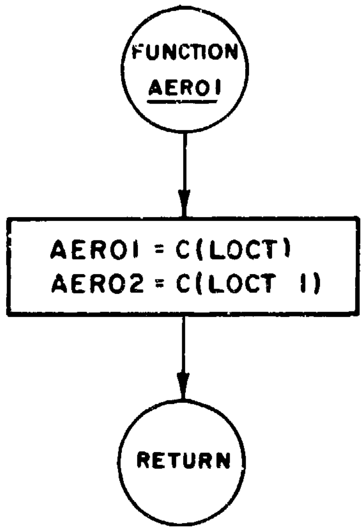
FLOW DIAGRAM (SACSI)



FLOW DIAGRAM (SACSI)



FLOW DIAGRAM (AERO1)



where A = the coefficient of $|X|X$

B = the coefficient of X

C = the constant term

XLLIM = the lower limit of X

XULIM = the upper limit of X

After the function is executived, the value of X will be stored in AQUAD.

27. OPTI - Six-Degree -Of-Freedom Trajectory Program Over a Flat Planet

a. Purpose - This program permits the computation of the six rigid-body degrees-of-freedom of a flight vehicle where the motion is assumed to occur over a limited portion of the planet. Under this assumption, the motion is taken to occur within a rectangular coordinate system. This program of computation has been provided to aid in such analyses as:

(1) Boost-phase dynamics.

(2) Pitch-roll-yaw coupling motion investigations.

(3) Autopilot response to parametric disturbance.

(4) Landing, approach, and flare-out maneuvers, etc.

(5) Landing gear dynamics.

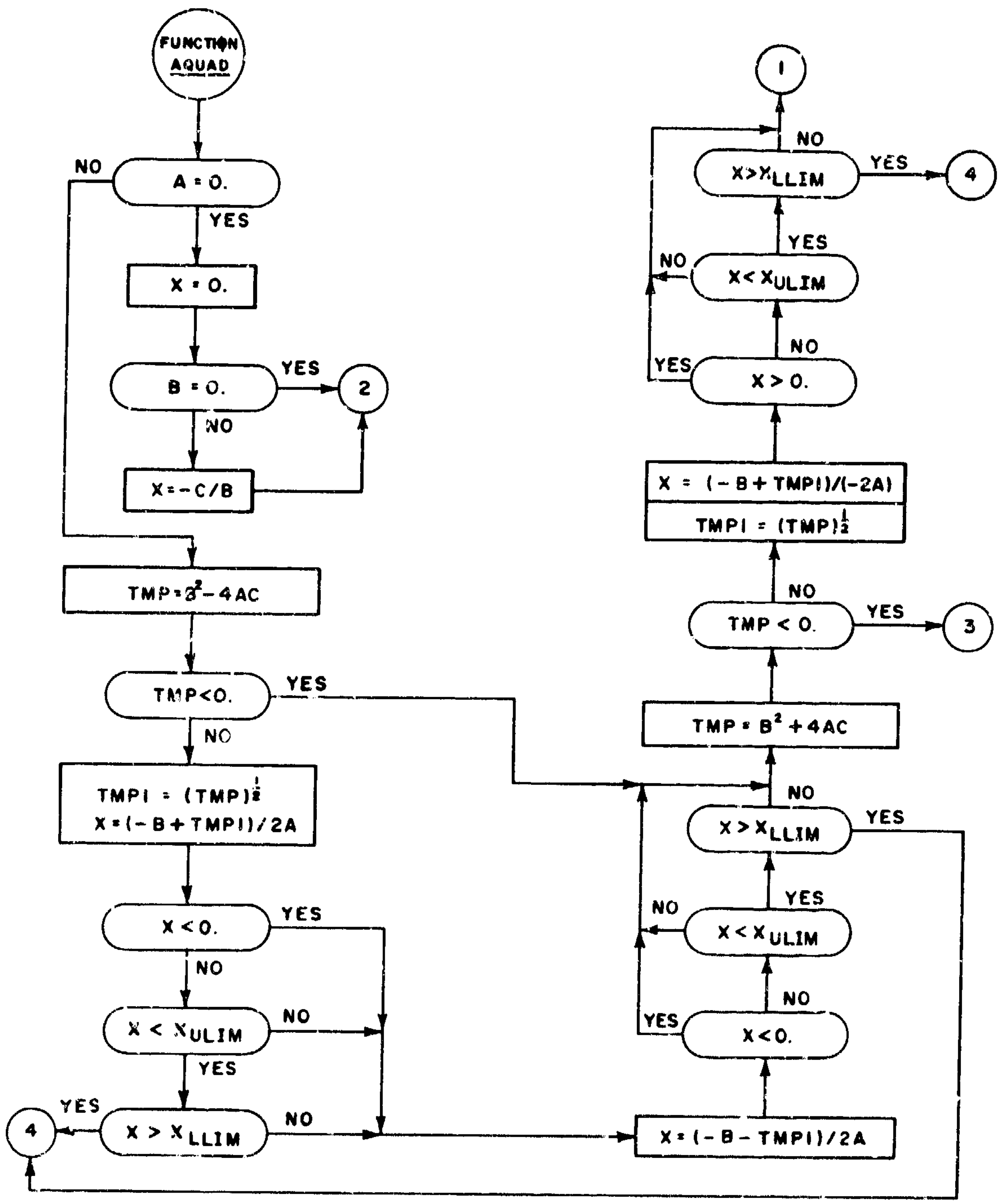
The effects upon the motion of the body of the following items may also be accounted for:

(1) Three components of wind velocity and acceleration measured in a local-geocentric Cartesian coordinate system.

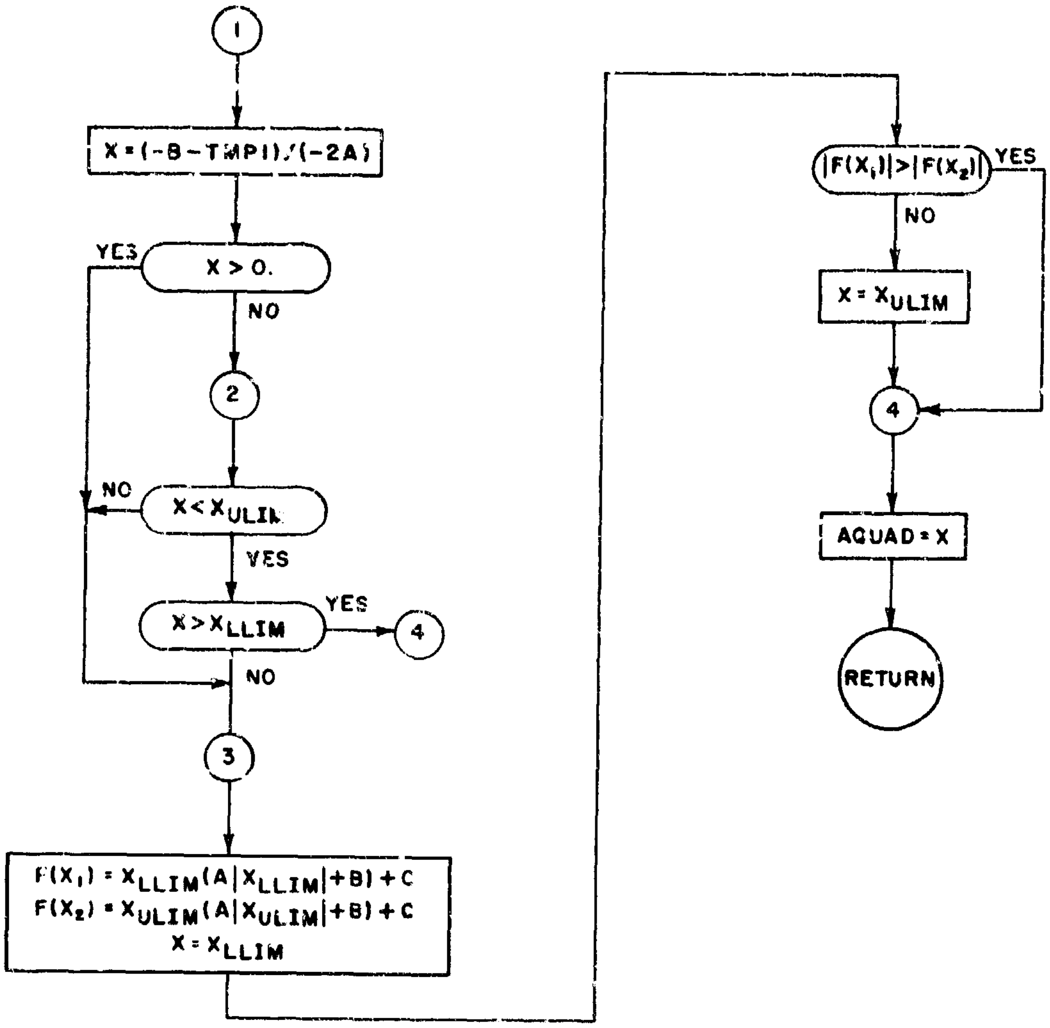
(2) An arbitrary planetary atmosphere.

(3) The gyroscopic torques imposed by rotating machinery aboard the vehicle.

FLOW DIAGRAM (AQUAD.)



FLOW DIAGRAM (AQUAD)



(4) The action of an arbitrary* three-plane autopilot upon the control deflections of the flight vehicle in response to the measured motion. The motion may be obtained from a stabilized platform. In this program the motion is more conveniently printed out in the Cartesian Coordinates.

b. Linkage

(1) CALL ØPT1

Pre-data initialization.

(2) CALL ØPT2

Initialization after data is read in and computation for initial time point.

(3) CALL ØPT4

Calculation of equation of motion, and all other variables that are desired.

(4) CALL ØPT6

Time history printout.

(5) CALL ØPT7

Update variables that are integrated by the integration routine.

28. LGEAR1 - LANDING GEAR CALCULATIONS, PART I

a. Purpose - The LGEAR1 subroutine computes the effects of ground reaction and landing gear dynamics on the motion of a landing vehicle. A maximum of five independent landing gears may be used.

b. Usage - Linkage to LGEAR1 is provided by the following statements:

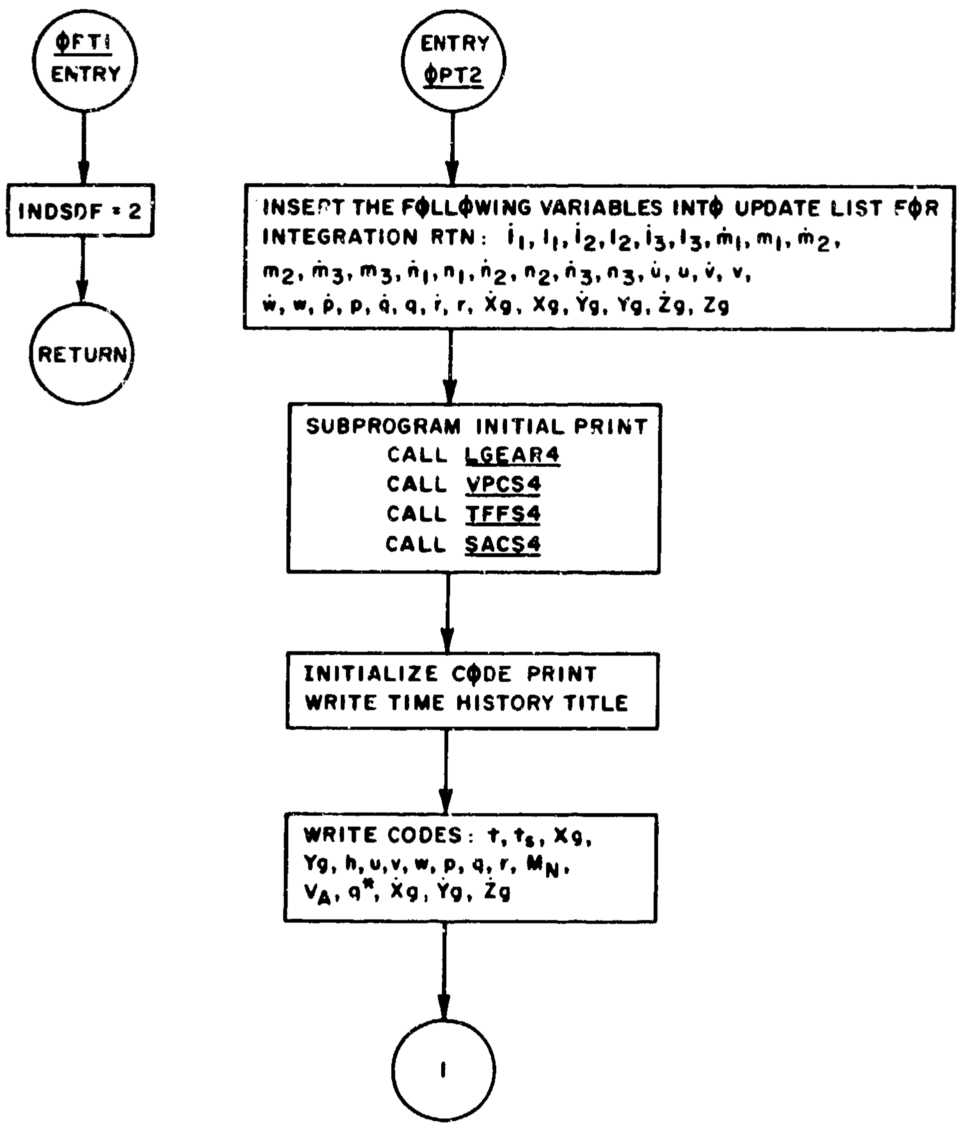
(1) CALL LGEAR1

Pre-data initialization.

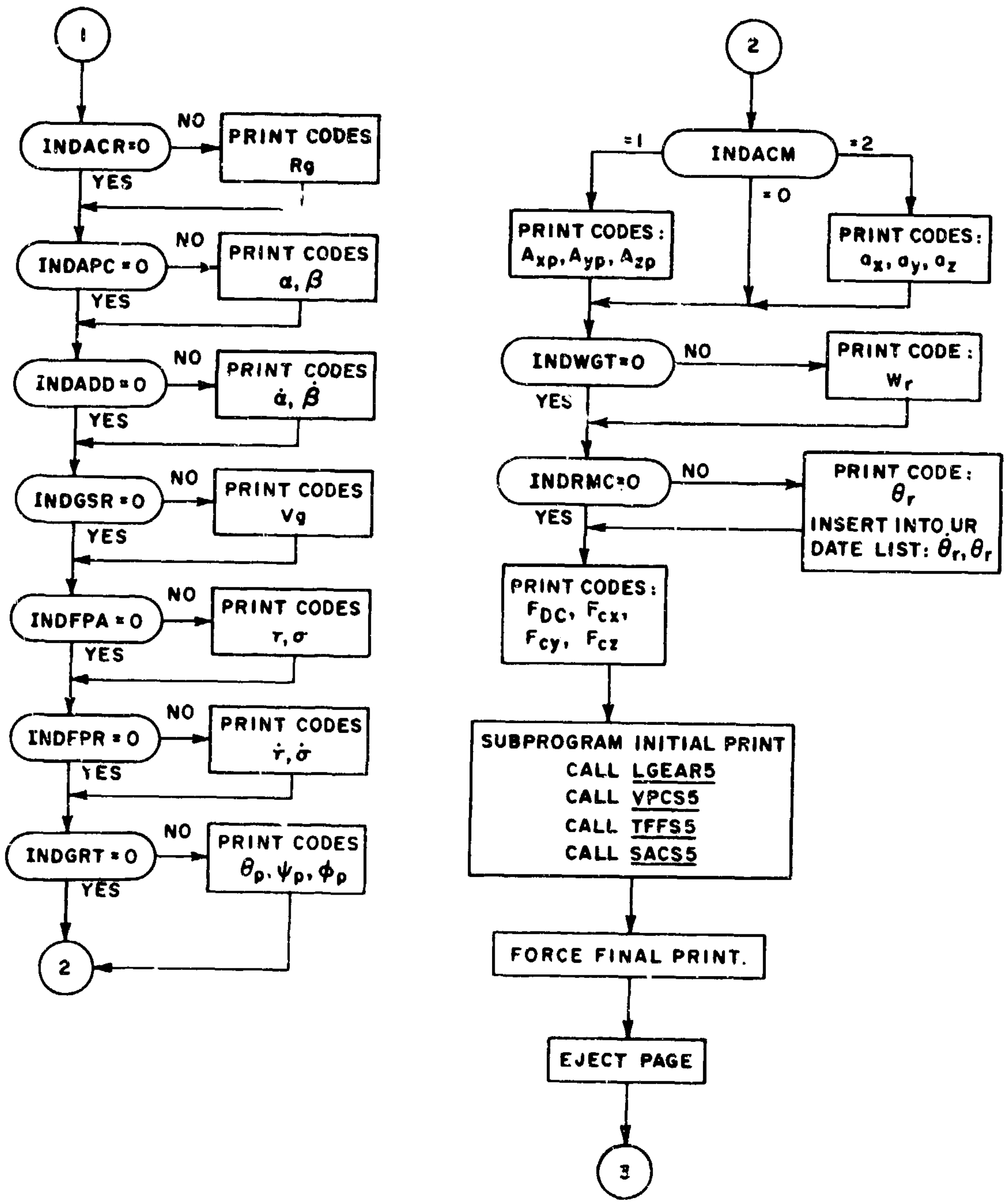
---

* Currently, only a Pitch-Yaw-Roll sequence is programmed, due to storage limitations.

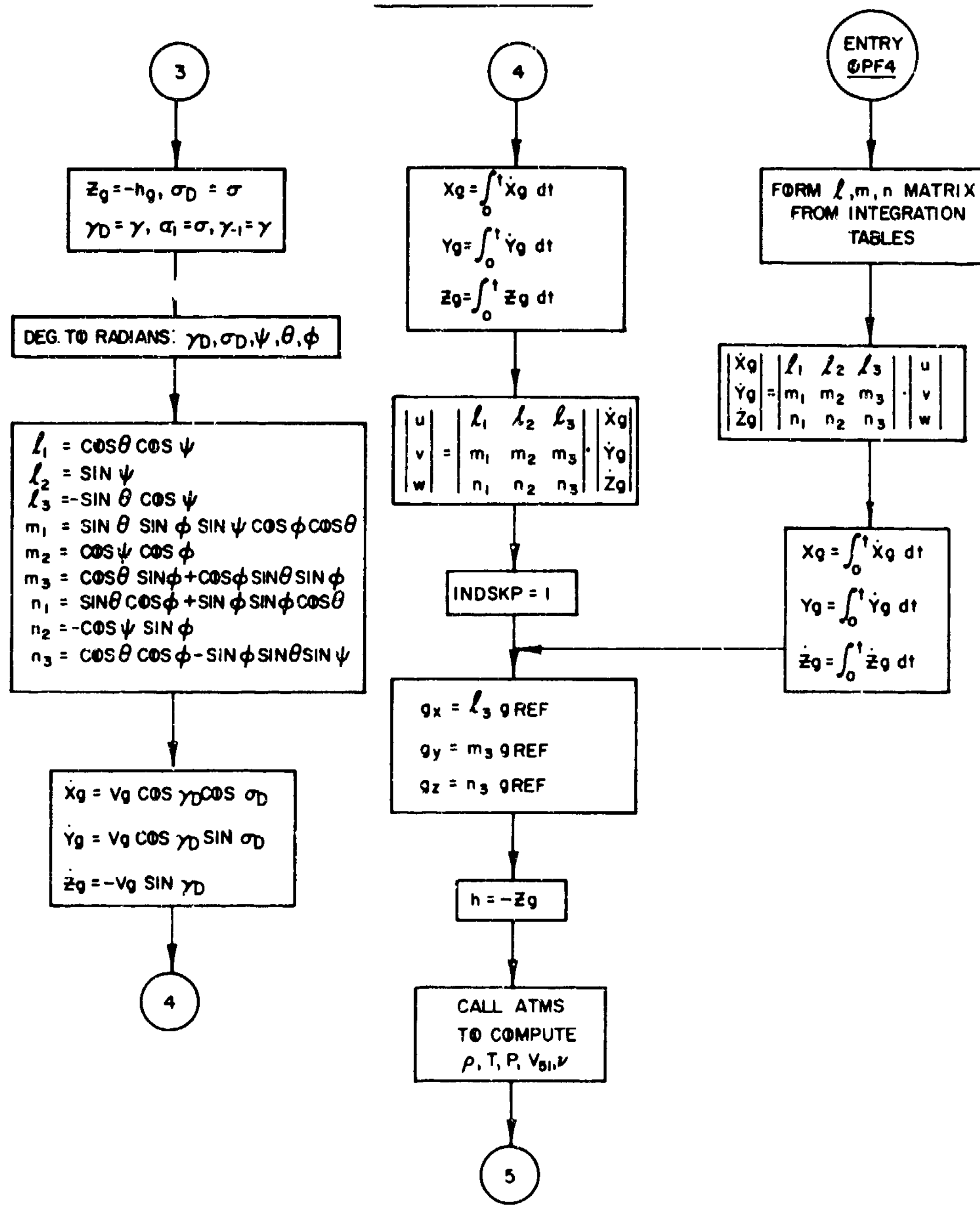
FLOW DIAGRAM (ΦPT1)



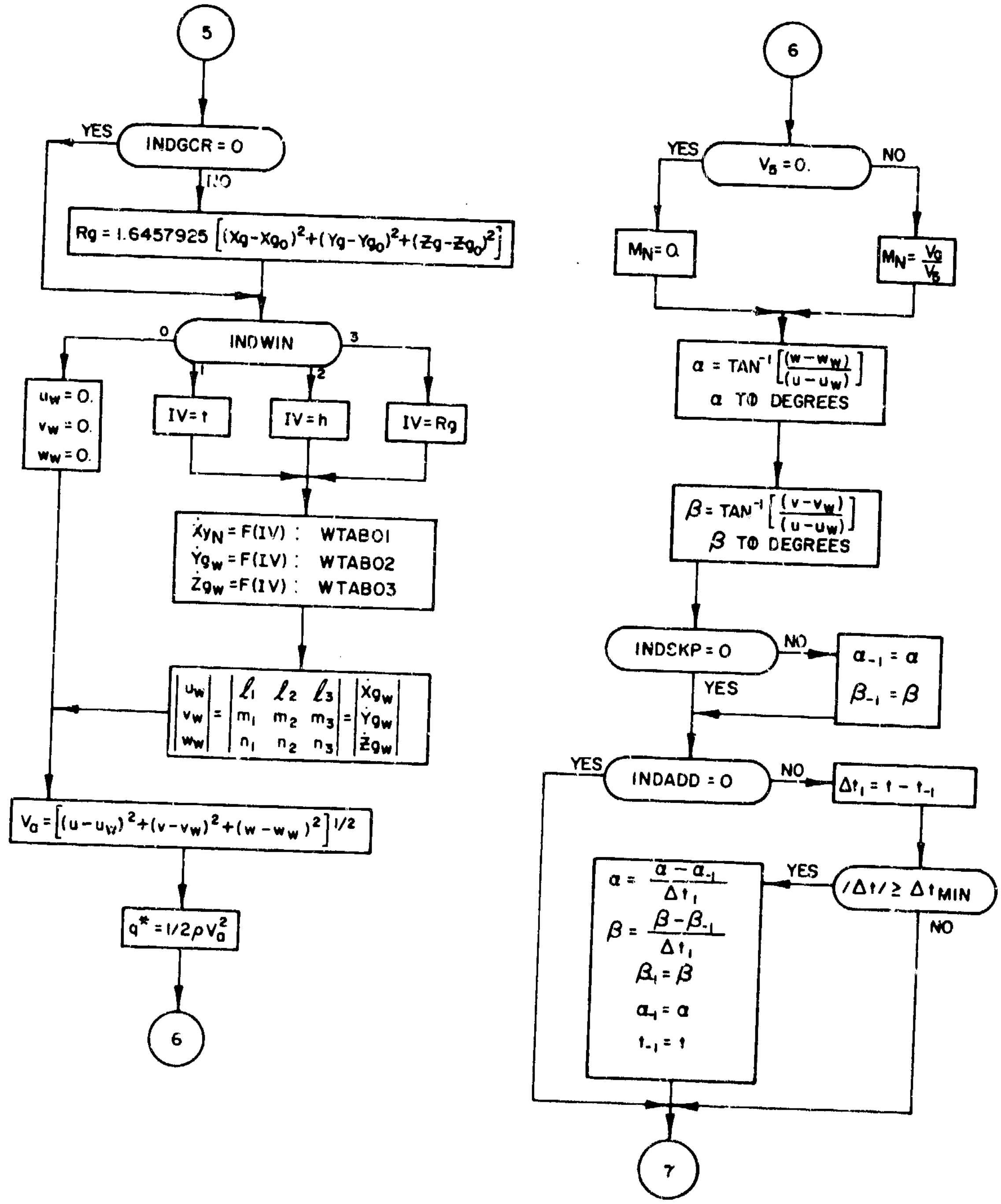
FLOW DIAGRAM (ΦFTI)



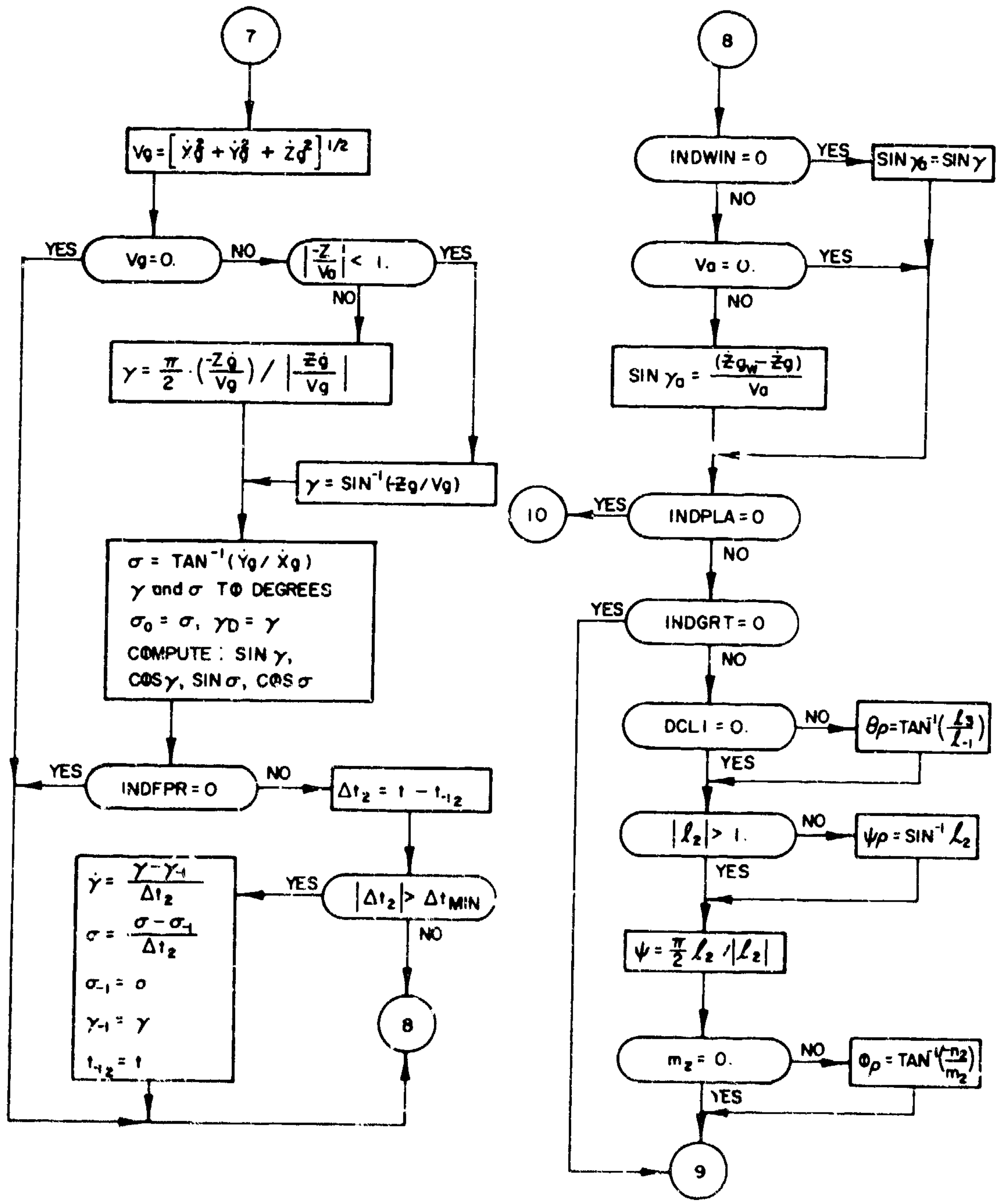
FLOW DIAGRAM (OPT1)



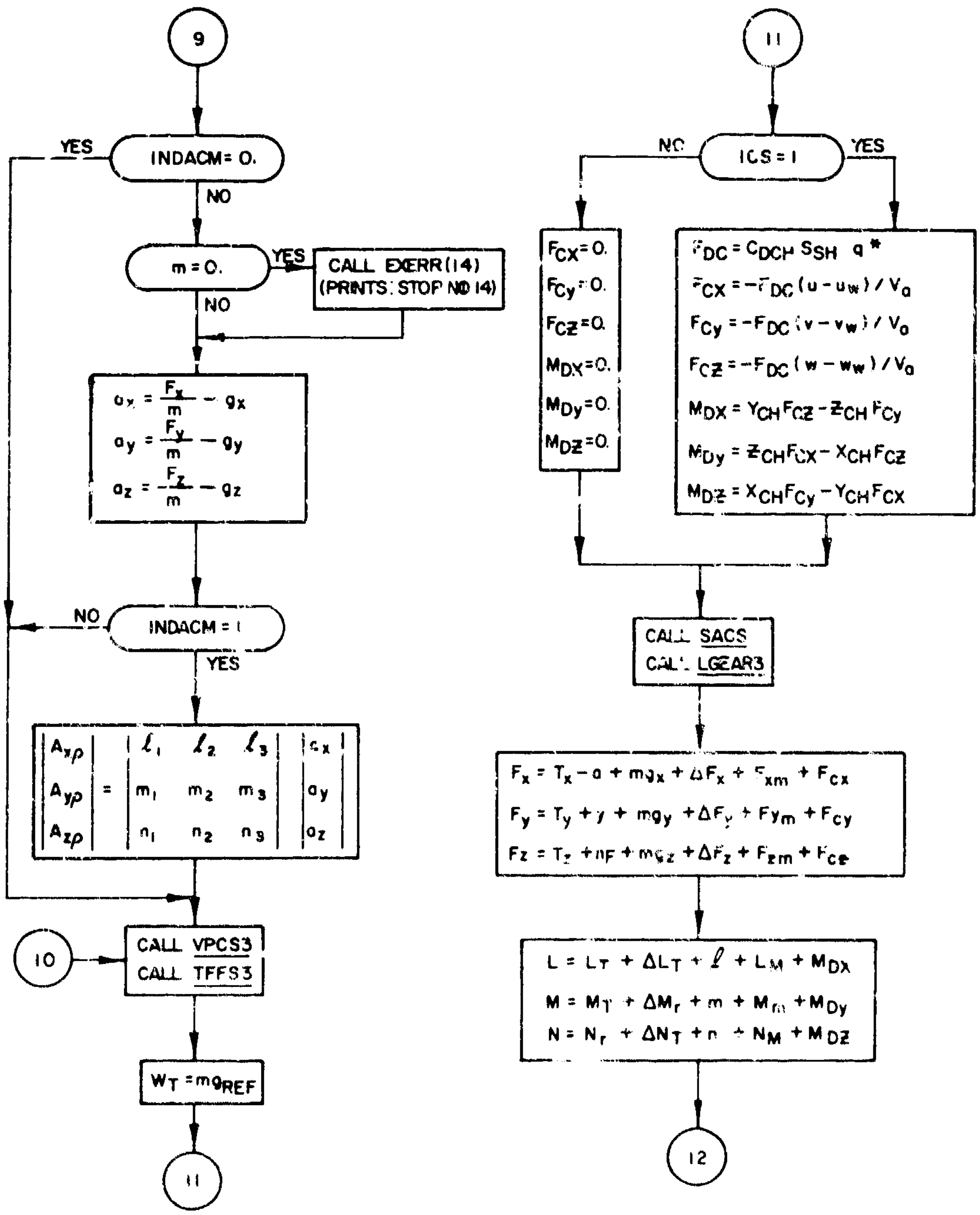
FLOW DIAGRAM (OPT1)



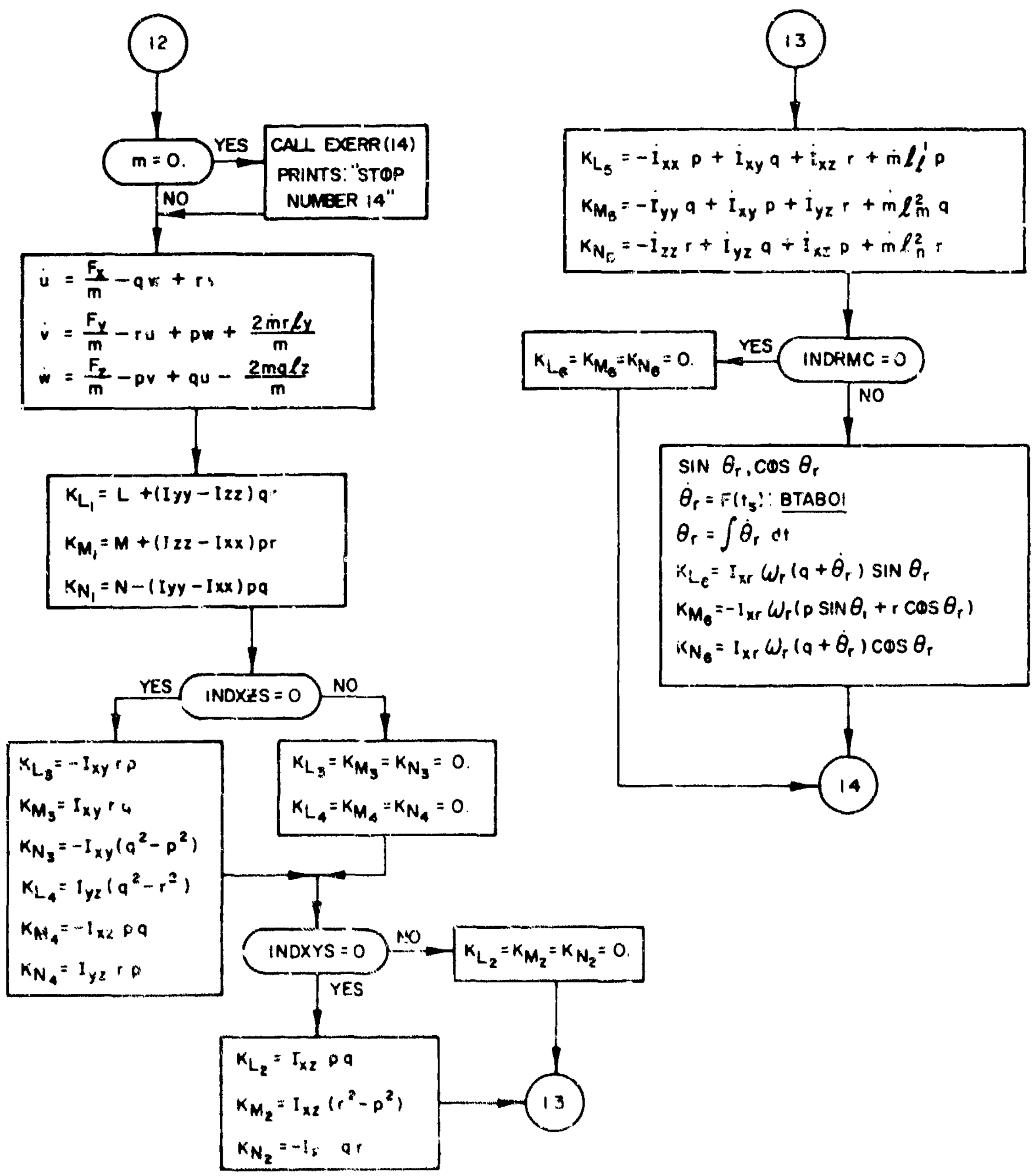
FLOW DIAGRAM (OPTI)



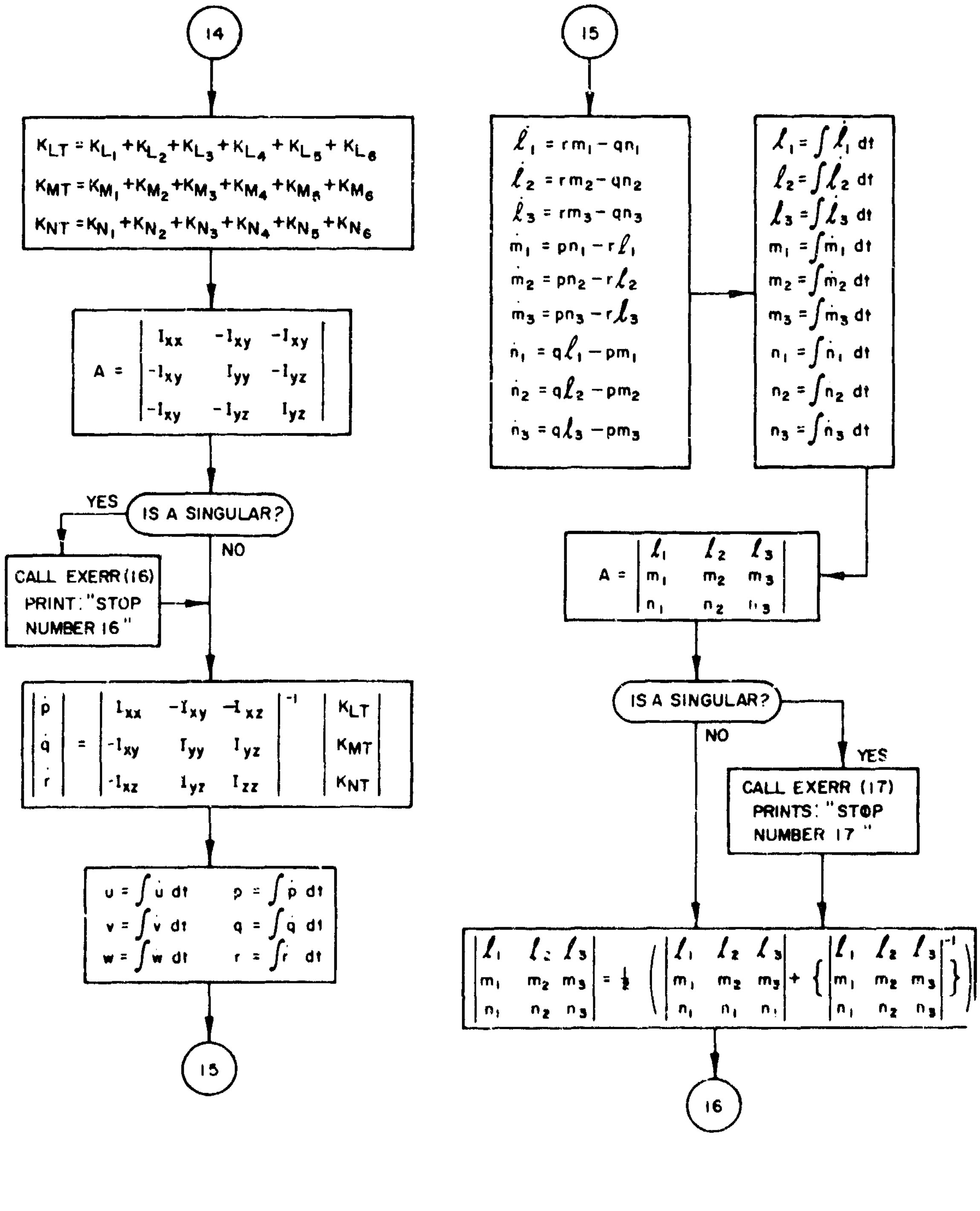
FLOW DIAGRAM (OPT1)



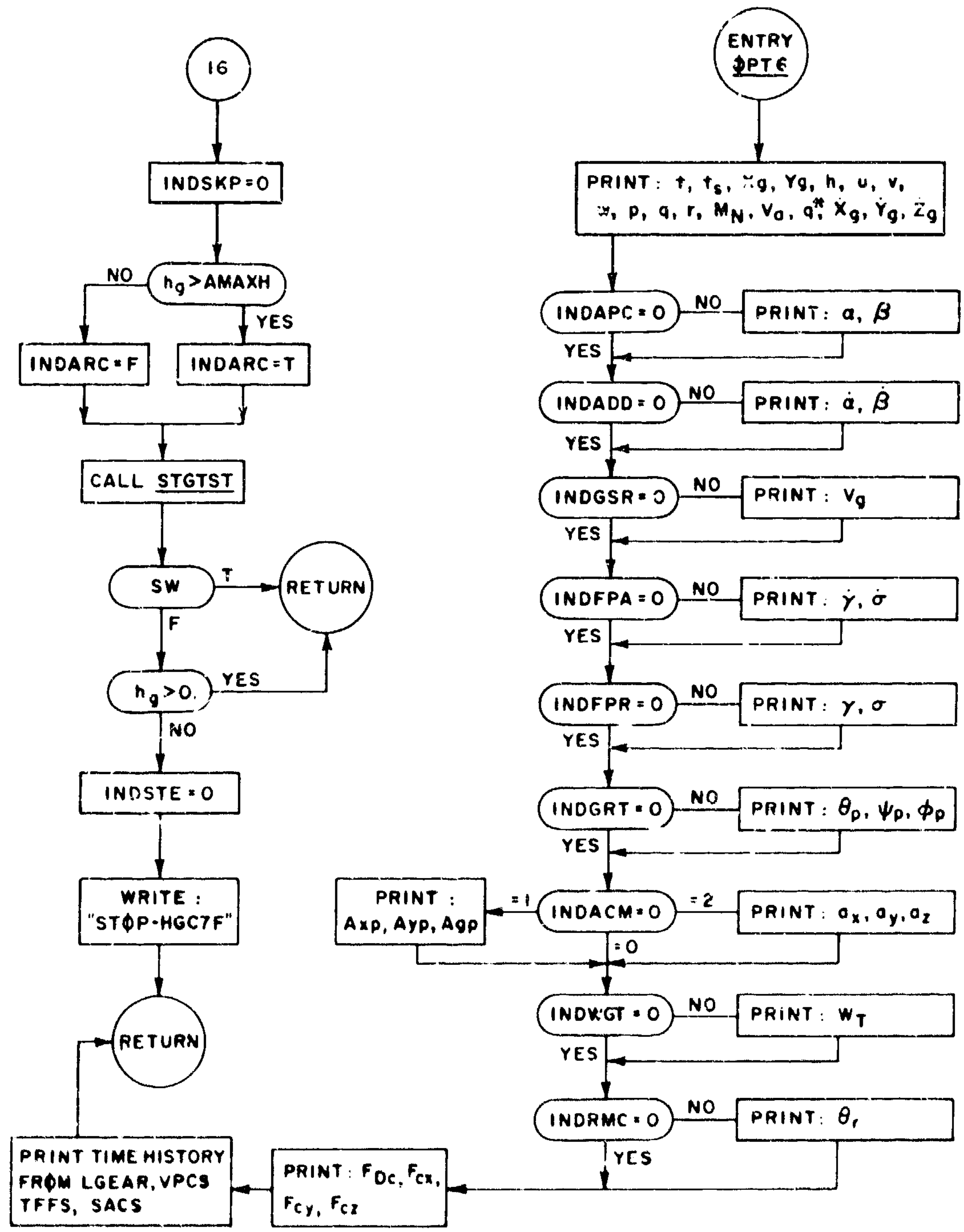
FLOW DIAGRAM (OPT1)



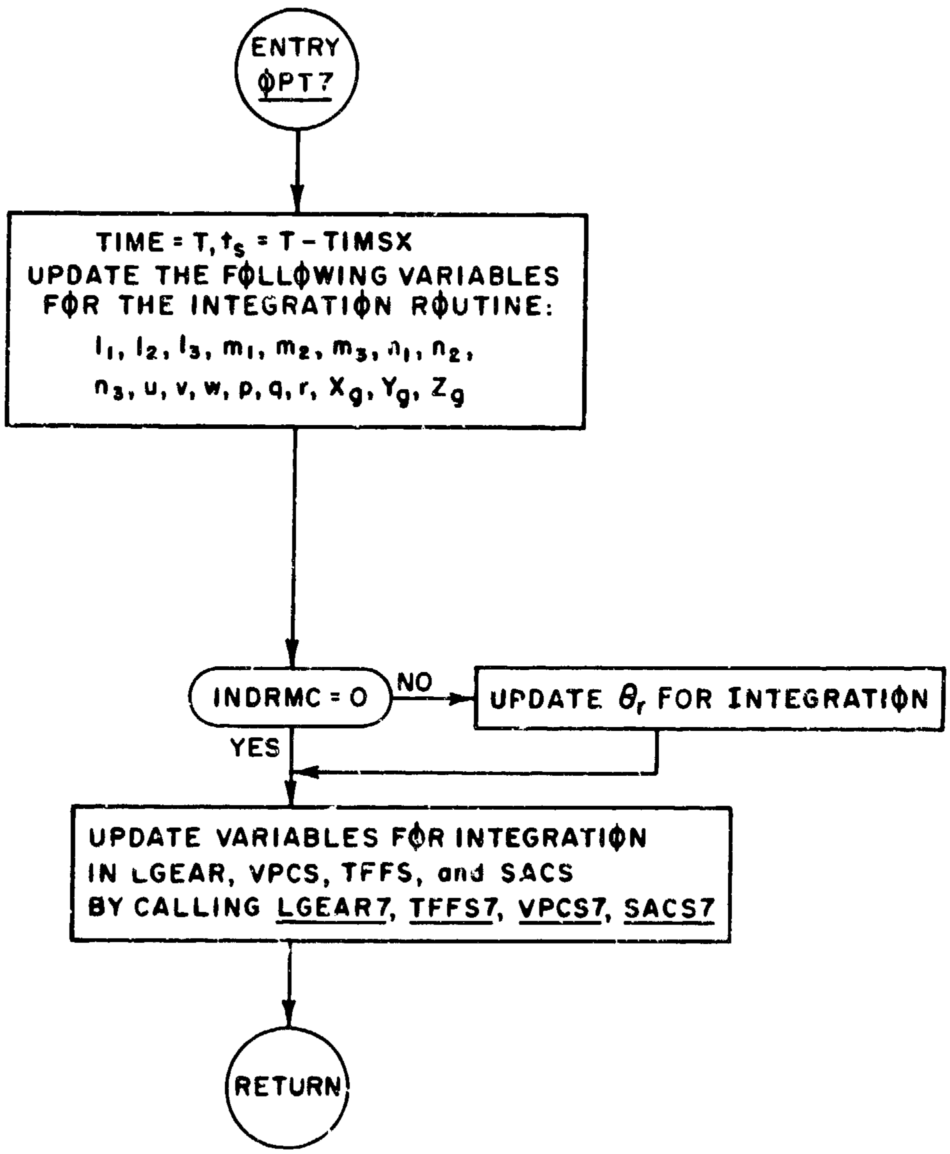
FLOW DIAGRAM (OPTI)



FLOW DIAGRAM (ΦPTI)



FLOW DIAGRAM (ϕPT7)



(2) CALL LGEAR2

Initialization after data read in.

(3) CALL LGEAR3

Landing gear dynamics computation, Part I

(4) CALL LGEAR4

Initial Print. (None in this subroutine).

29. LGEA3C - LANDING GEAR CALCULATIONS, PART II

a. Purpose - Continuation of the calculations of the effects of ground reaction and landing gear dynamics on the motion of a landing vehicle.

b. Usage - Linkage to LGEA3C is provided by the following statement:

CALL LGEA3C

30. SDFLGP - PRINTING OF SDF AND LG VARIABLES

a. Purpose - The SDFLGP subroutine saves data on tape for plotting, prints output, and updates variables for the integration routine.

b. Usage - Linkage to SDFLGP is provided by the following statements.

(1) CALL SDFLGP

Saves data on tape for plotting.

Input cards required:

Column:	1	12
	IPLT	1
	ISDF	1
	ISTPL1	1
	ISTPL2	1
	ISTPL3	1
	ISTPL4	1
	ISTPL5	1

IPLT = 1 denotes that data will be saved on tape unit 13 for plotting. ISDF=1 denotes that rigid body data will be saved on tape. ISTPL denotes data for landing gear k will be saved on tape.

(2) CALL LGEA6P

Writes code on tape to designate last record for a particular case and also writes EOF.

(3) CALL LGEAR6

Prints output of the Landing Gear calculations

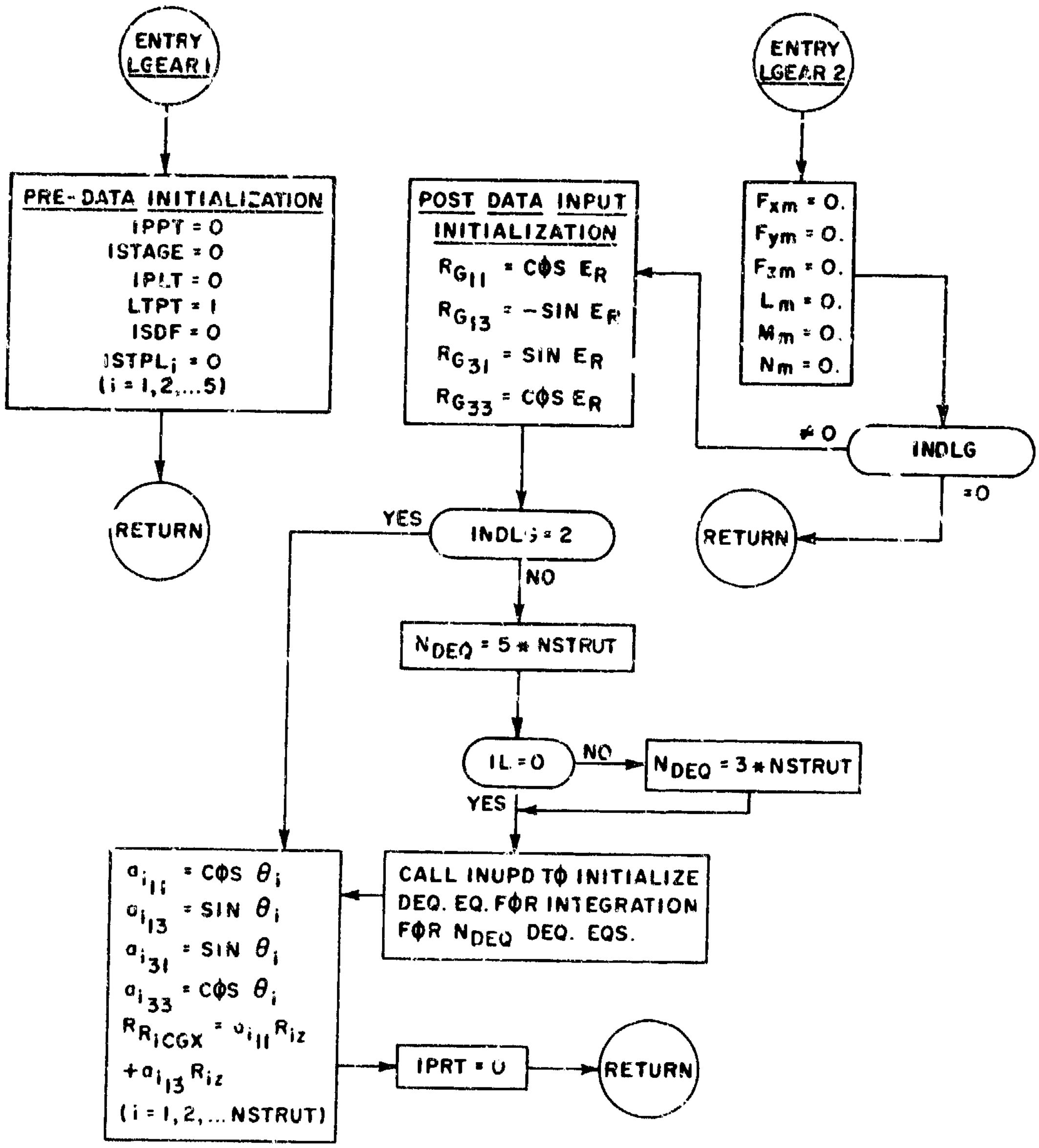
(4) CALL LGEAR7

Updates landing gear variables that are integrated by the integration routine.

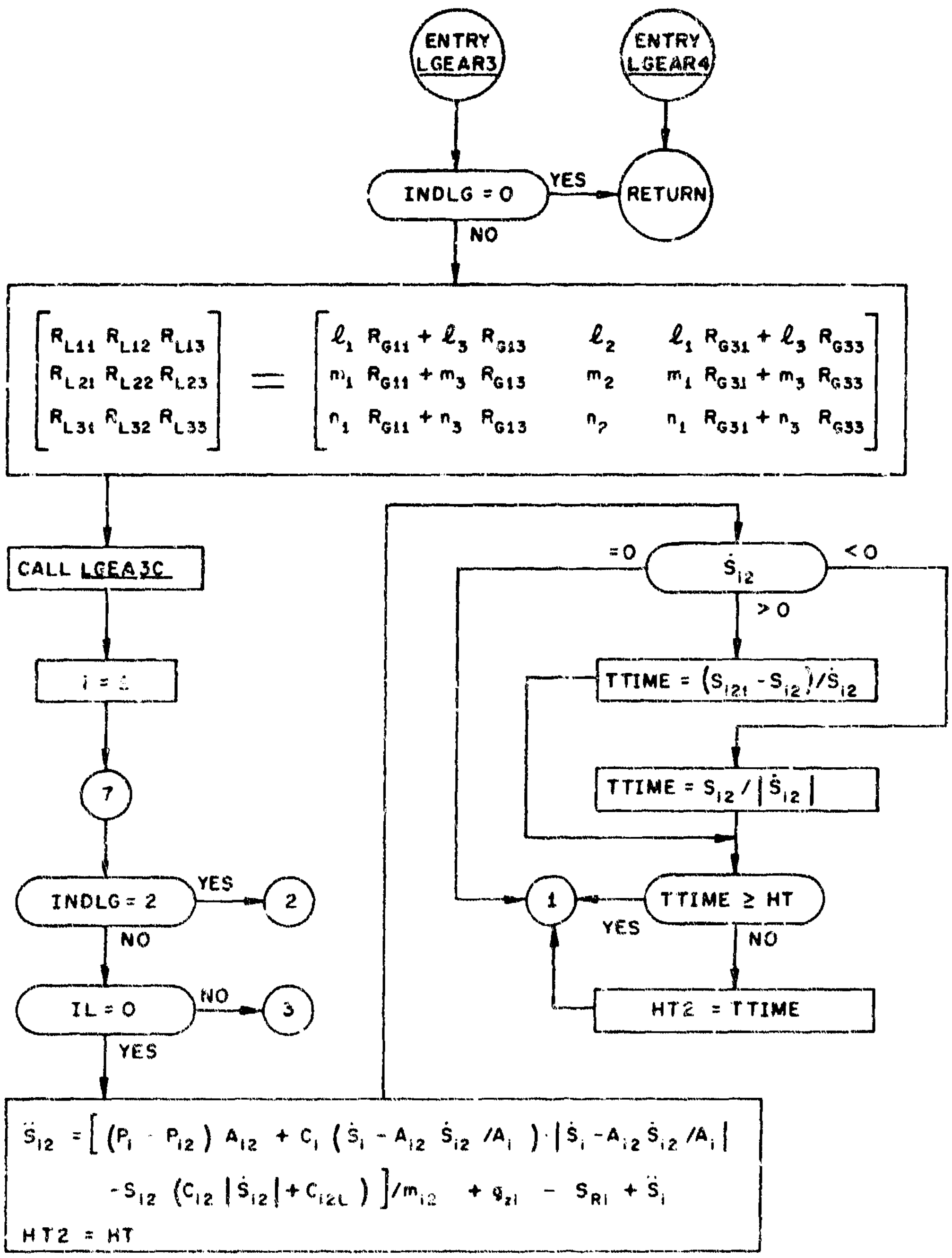
(5) CALL LGEAR5

Prints titles. (Not used, only a return statement.)

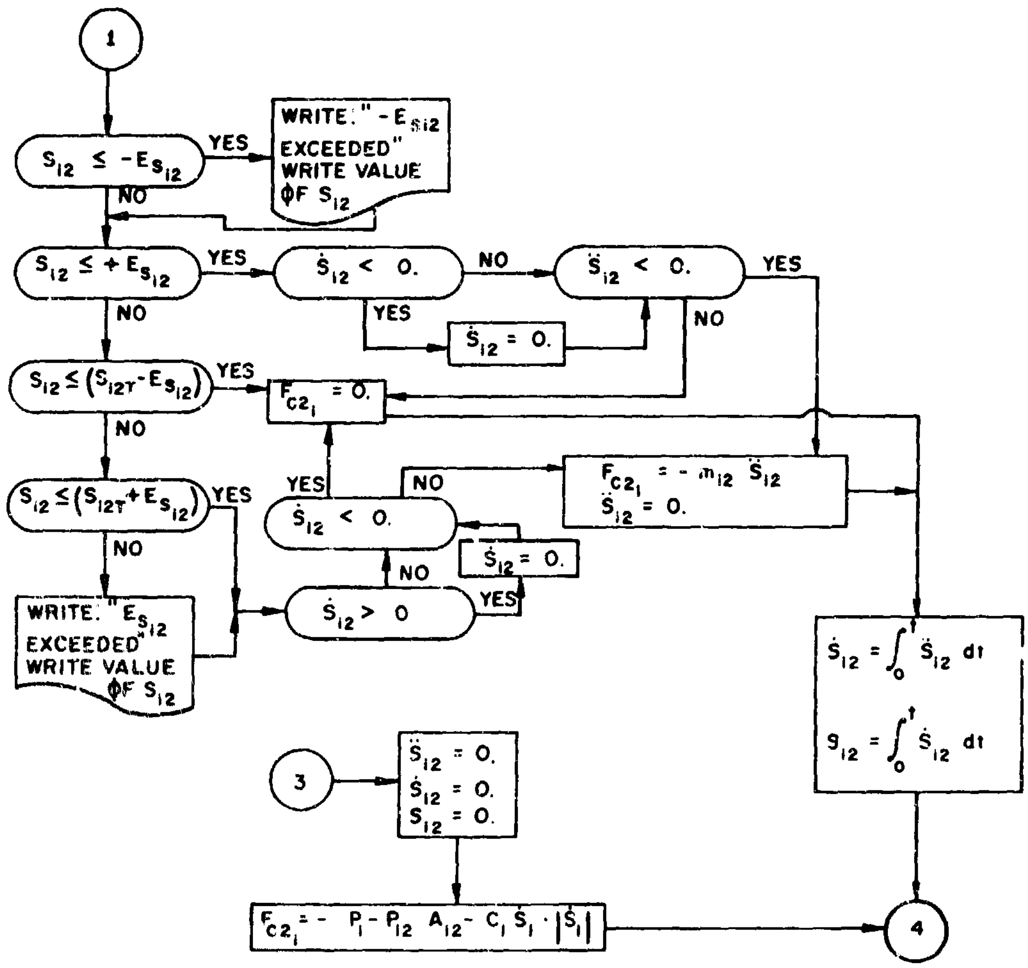
FLOW DIAGRAM (LGEAR1)



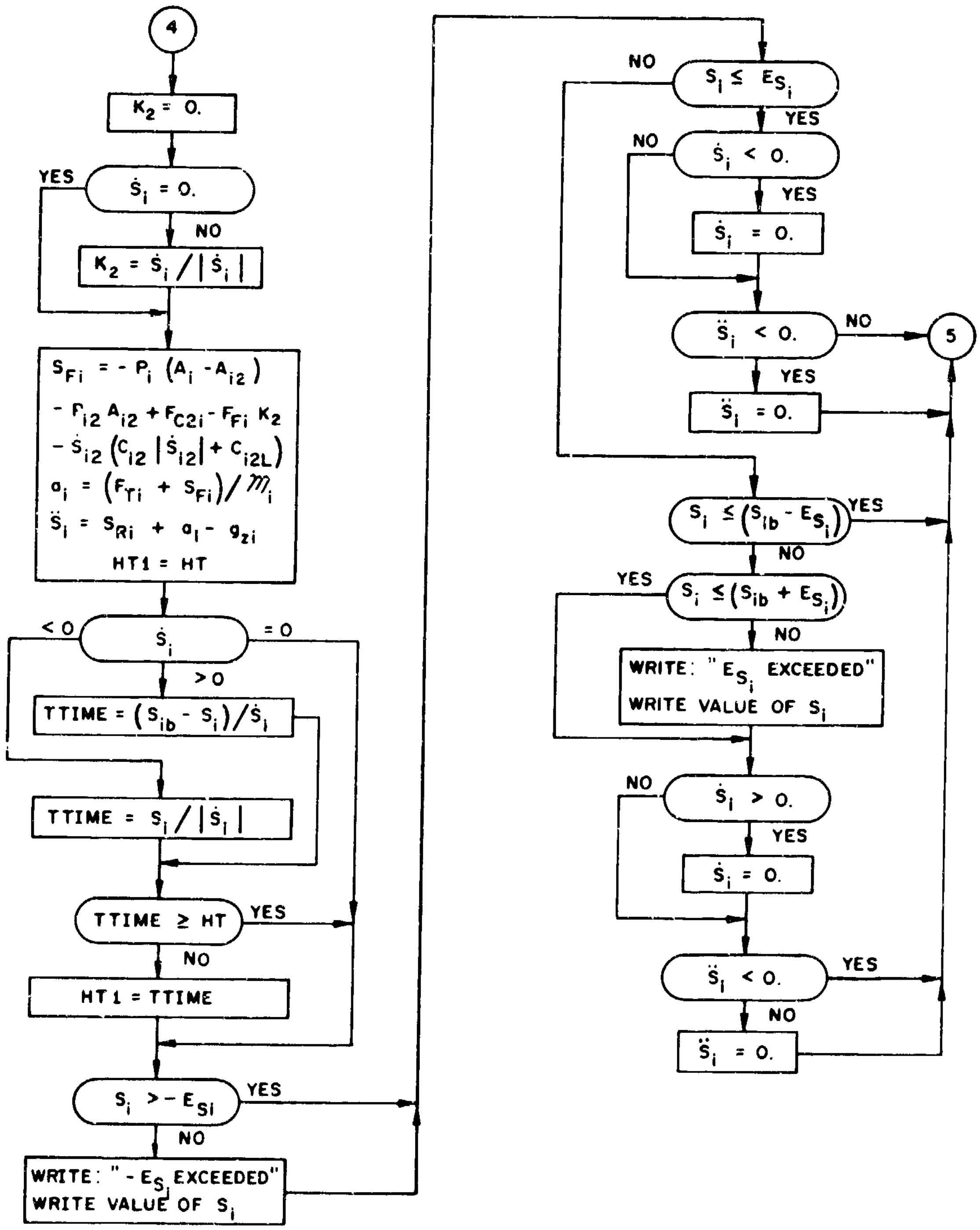
FLOW DIAGRAM (LGEAR1)



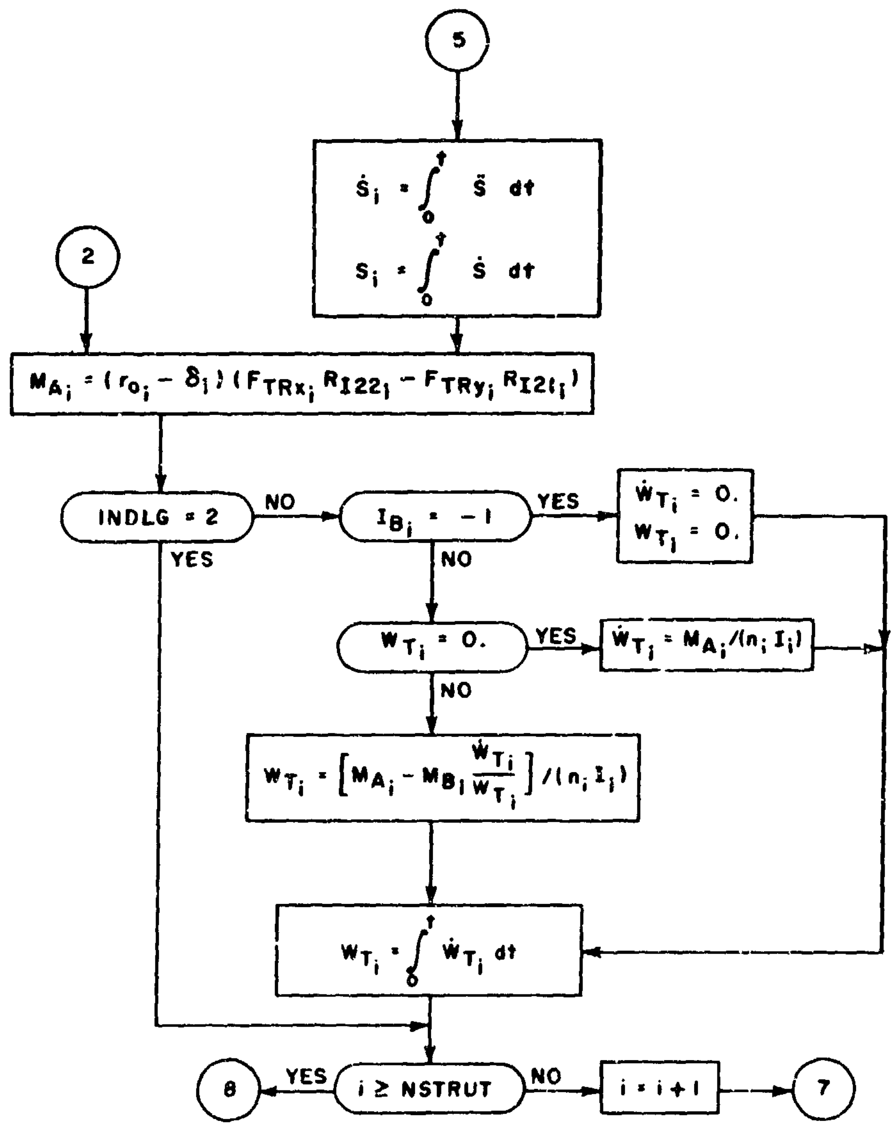
FLOW DIAGRAM (LGEAR1)



FLOW DIAGRAM (LGEAR1)



FLOW DIAGRAM (LGEAR1)



FLOW DIAGRAM (L GEAR I)

8

$$F_{TRX} = \sum_{i=1}^{NSTRUT} F_{TRX_i}, \quad F_{TRY} = \sum_{i=1}^{NSTRUT} = F_{TRY_i}, \quad F_{TRZ} = \sum_{i=1}^{NSTRUT} F_{TRZ_i}$$

$$\begin{bmatrix} F_{TRA} \\ F_{TRB} \\ F_{TRC} \end{bmatrix} = \begin{bmatrix} R_{L11} & R_{L12} & R_{L13} \\ R_{L21} & R_{L22} & R_{L23} \\ R_{L31} & R_{L32} & R_{L33} \end{bmatrix} \cdot \begin{bmatrix} F_{TRX} \\ F_{TRY} \\ F_{TRZ} \end{bmatrix}$$

$$M_{TRX} = \sum_{i=1}^{NSTRUT} M_{TRX_i}, \quad M_{TRY} = \sum_{i=1}^{NSTRUT} M_{TRY_i}, \quad M_{TRZ} = \sum_{i=1}^{NSTRUT} M_{TRZ_i}$$

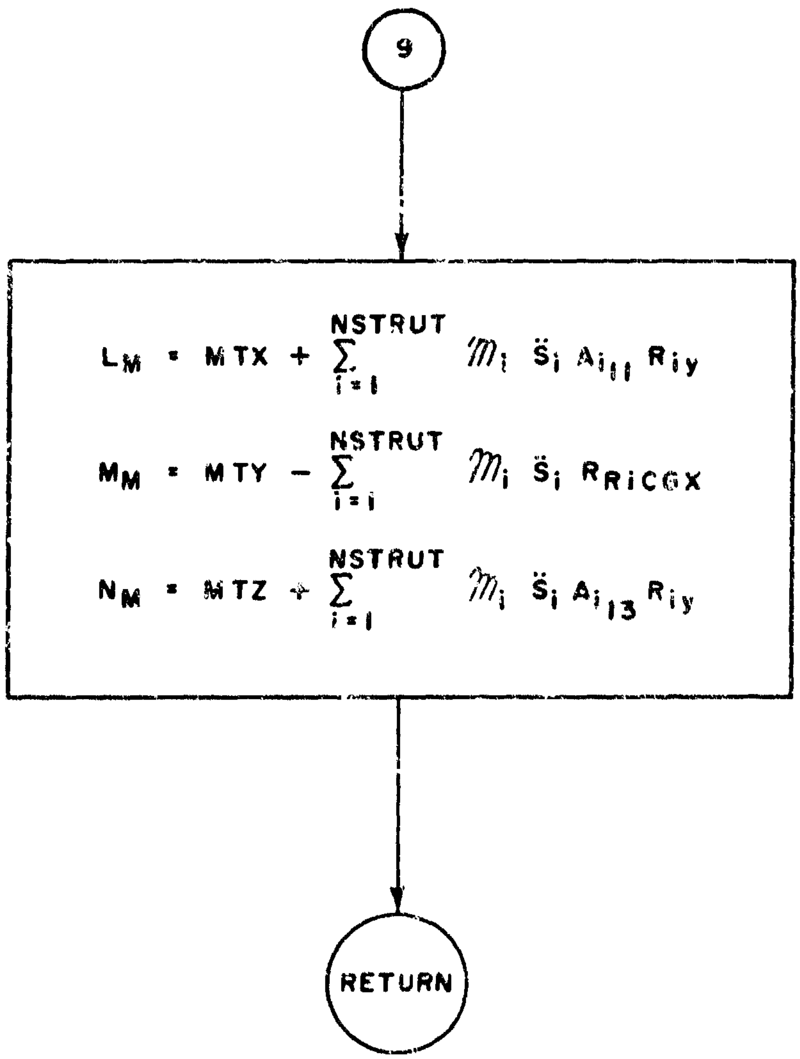
$$\begin{bmatrix} M_{TX} \\ M_{TY} \\ M_{TZ} \end{bmatrix} = \begin{bmatrix} R_{L11} & R_{L12} & R_{L13} \\ R_{L21} & R_{L22} & R_{L23} \\ R_{L31} & R_{L32} & R_{L33} \end{bmatrix} \cdot \begin{bmatrix} M_{TRX} \\ M_{TRY} \\ M_{TRZ} \end{bmatrix}$$

$$F_{XM} = F_{TRA} + \sum_{i=1}^{NSTRUT} m_i \ \ddot{s}_i \ a_{i31}, \quad F_{YM} = F_{TRB}$$

$$F_{ZM} = F_{TRC} + \sum_{i=1}^{NSTRUT} m_i \ \ddot{s}_i \ a_{i33}$$

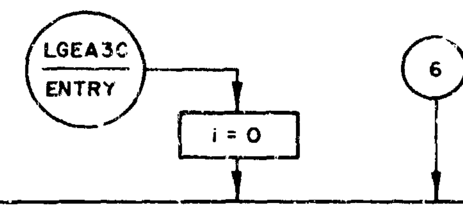
9

FLOW DIAGRAM (LGEAR I)



FLOW DIAGRAM LGEA3C

(i = 1, 2, ---- NSTRUT IN THIS SUBROUTINE)



i = i + 1

$$\begin{bmatrix} R_{I_{11i}} & R_{I_{12i}} & R_{I_{13i}} \\ R_{I_{21i}} & R_{I_{22i}} & R_{I_{23i}} \\ R_{I_{31i}} & R_{I_{32i}} & R_{I_{33i}} \end{bmatrix} = \begin{bmatrix} a_{i11} R_{L11} + a_{i13} R_{L31} & a_{i11} R_{L12} + a_{i13} R_{L32} & a_{i11} R_{L13} + a_{i13} R_{L33} \\ R_{L21} & R_{L22} & R_{L23} \\ a_{i31} R_{L11} + a_{i33} R_{L31} & a_{i31} R_{L12} + a_{i33} R_{L32} & a_{i31} R_{L13} + a_{i33} R_{L33} \end{bmatrix}$$

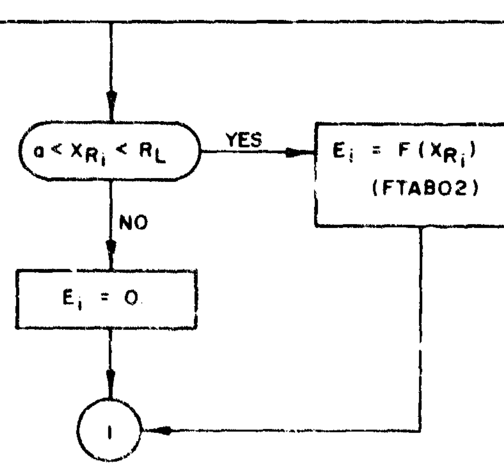
$$\begin{bmatrix} RAX_i \\ RAY_i \\ RAZ_i \end{bmatrix} = \begin{bmatrix} \ell_1 & m_1 & n_1 \\ \ell_2 & m_2 & n_2 \\ \ell_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} Rix + a_{i31} (rF_i - Si) \\ Riy \\ Riz + a_{i33} (rF_i - Si) \end{bmatrix}$$

$$Z_{0i} = (Xg - Rg_R + RAX_i)\, RG_{31} + (Zg + RAZ_i)\, RG_{33}$$

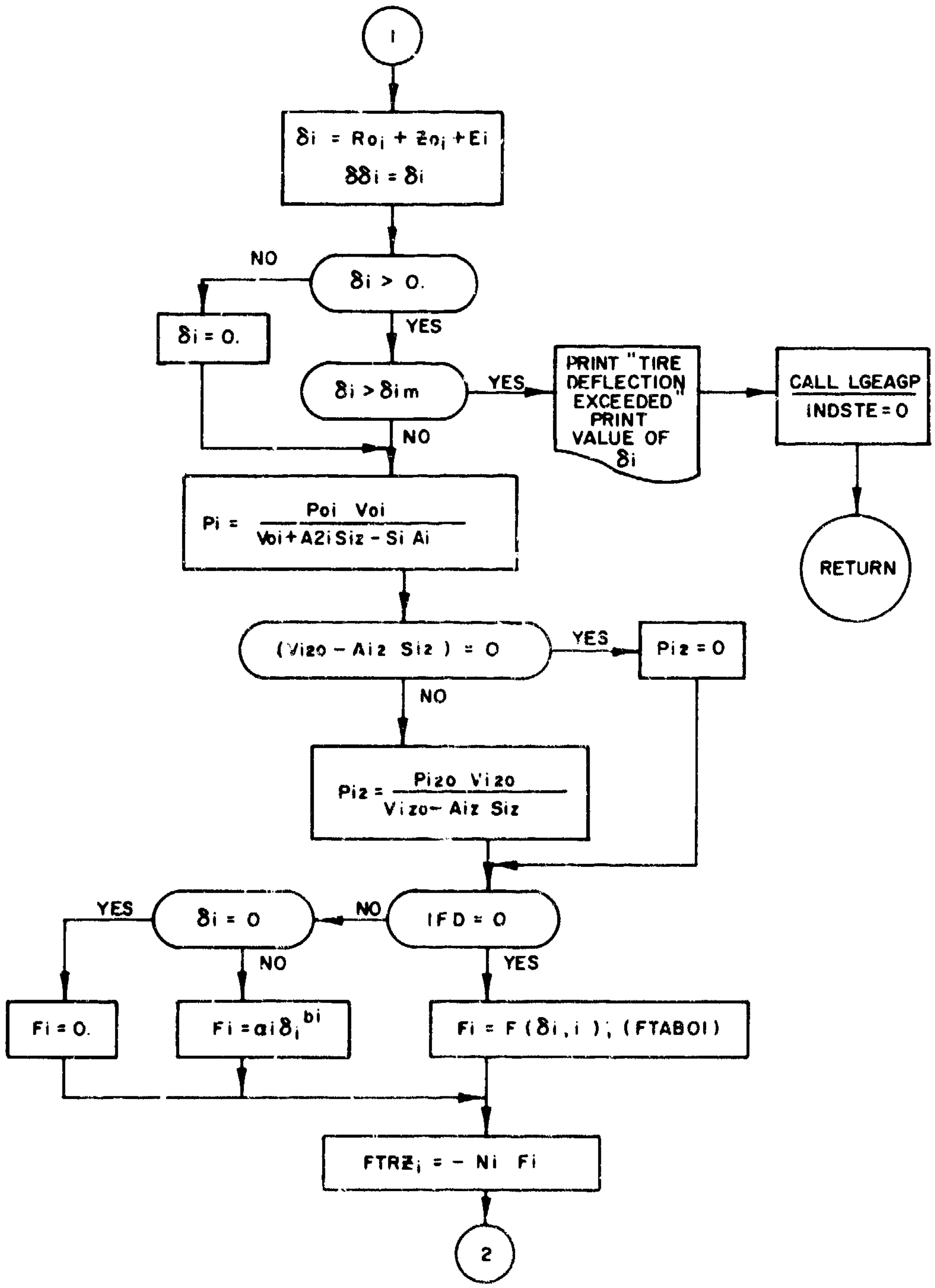
$$X_{R_i} = (Xg - Rg_R + RAX_i)\, RG_{11} + (Zg + RAZ_i)\, RG_{13}$$

$$Y_{R_i} = Yg + RAY_i$$

FOR i = 1, 2, -------, NSTRUT



FLOW DIAGRAM LGEA3C



FLOW DIAGRAM LGEA3C

2

$$R_{Dx_i} = -a_{i_{31}} + q\left\{(r_{F_i} - s_i)\, a_{i_{33}} + R_{i_z}\right\} - r R_{i_y}$$

$$R_{Dy_i} = (r_{F_i} - s_i)(r a_{i_{31}} - P a_{i_{33}}) + r R_{i_x} - P R_{i_z}$$

$$R_{Dz_i} = -s_i\, a_{i_{33}} - (r_{F_i} - s_i)\, q a_{i_{31}} + P R_{i_y} - q R_{i_x}$$

$$\begin{bmatrix} R_{DXG_i} \\ R_{DYG_i} \\ R_{DZG_i} \end{bmatrix} = \begin{bmatrix} \dot{X}g \\ \dot{Y}g \\ \dot{Z}g \end{bmatrix} + \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} * \begin{bmatrix} R_{DX_i} \\ R_{DY_i} \\ R_{DZ_i} \end{bmatrix}$$

$$VTX_i = RG_{11}\, RDXG_i + RG_{13}\, RDZG_i + WT_i\, RI_{22i}\,(ro_i - \delta_i)$$

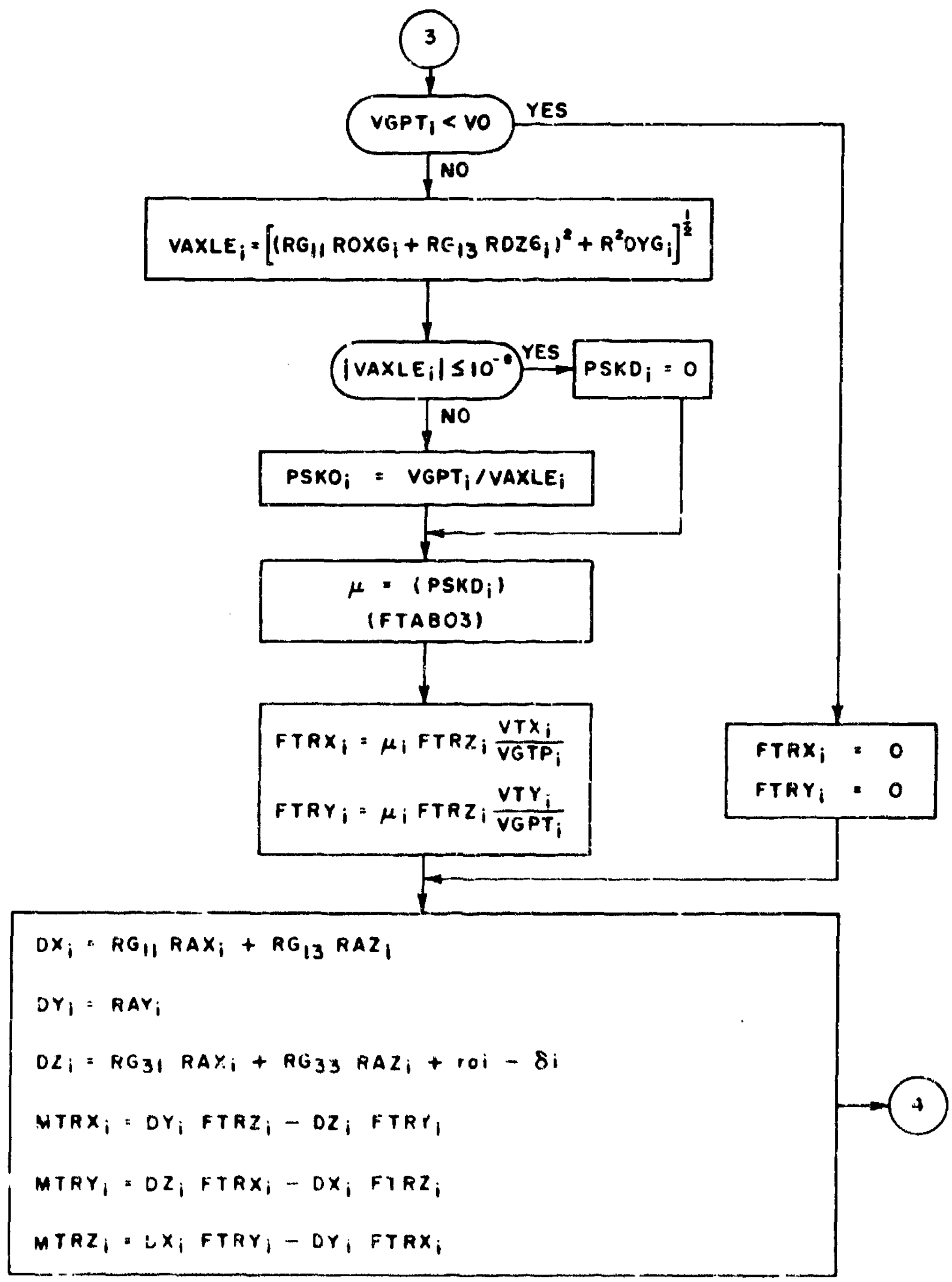
$$VTY_i = ROYG_i - WT_i\, RI_{21i}\,(ro_i - \delta_i)$$

$$VTZ_i = RG_{31}\, RDXG_i + RG_{33}\, RDZG_i$$

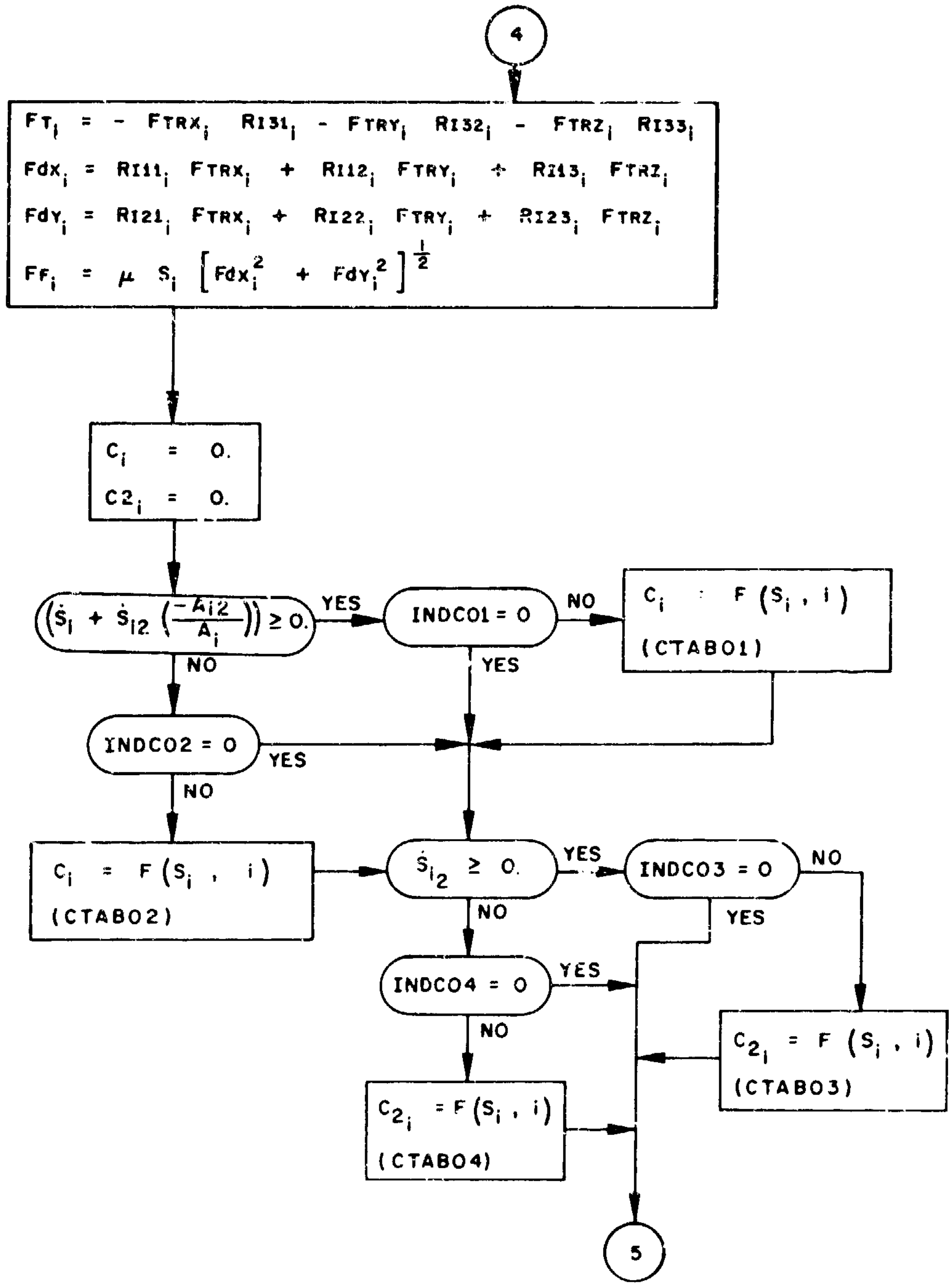
$$VGPT_i = (V^2TX_i + V^2TY_i)^{\frac{1}{2}}$$

3

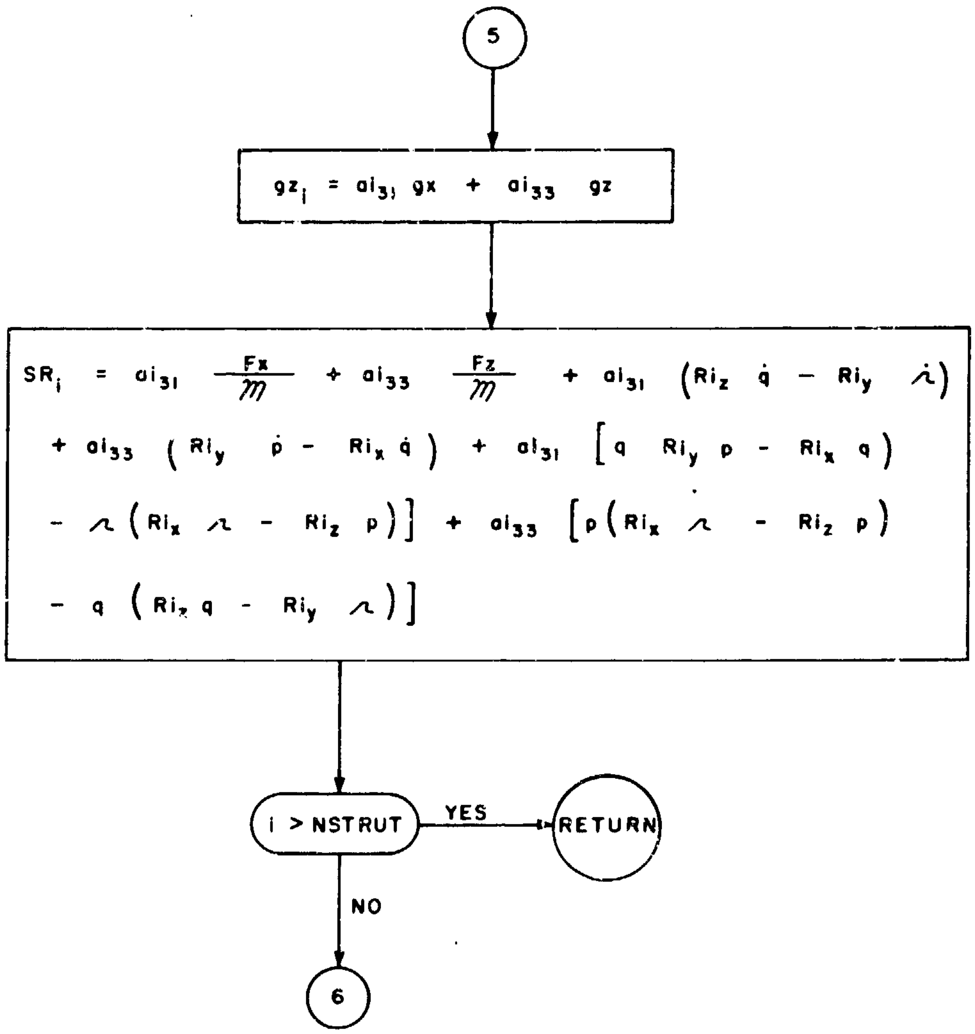
FLOW DIAGRAM LGEA3C



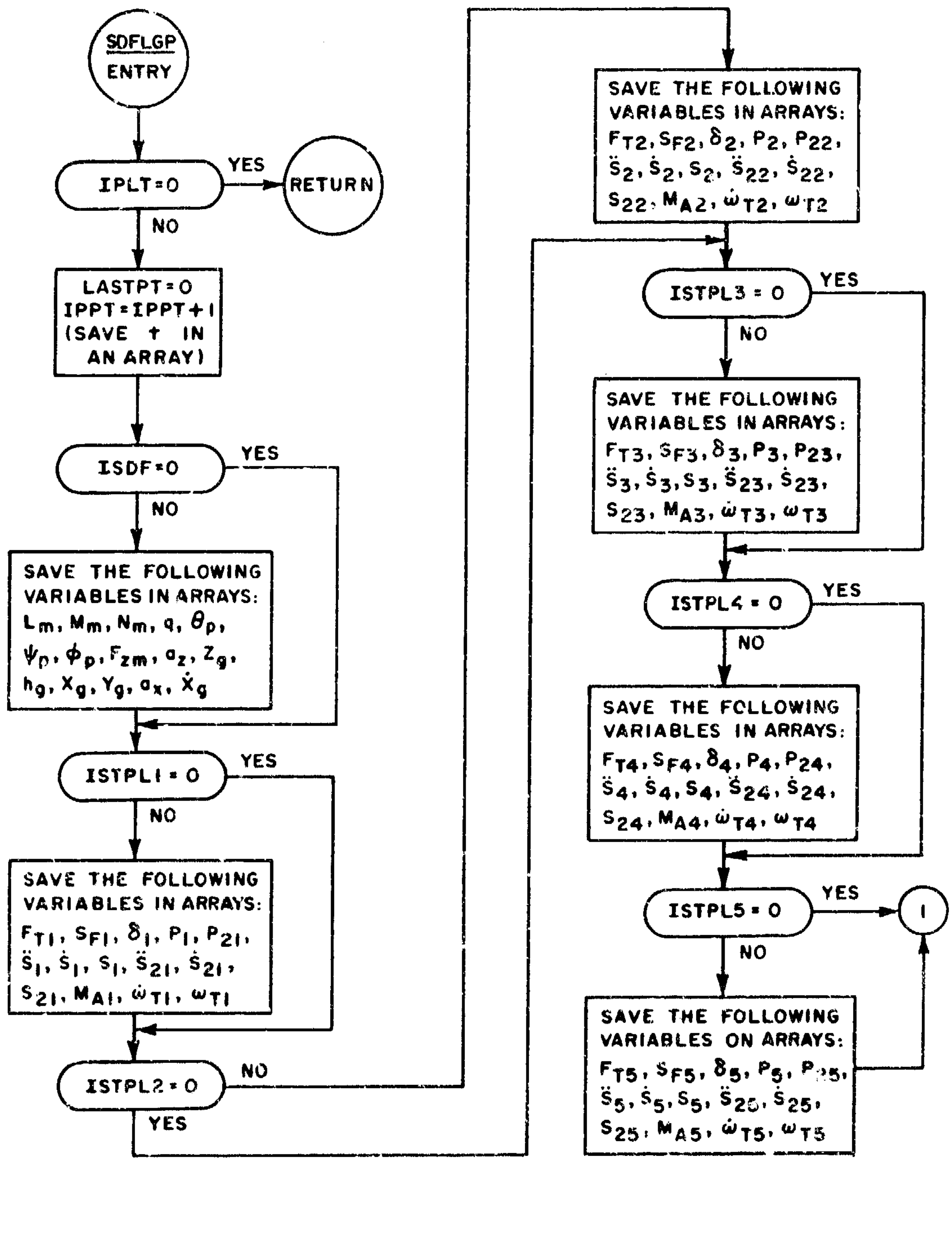
FLOW DIAGRAM, LGEA3C



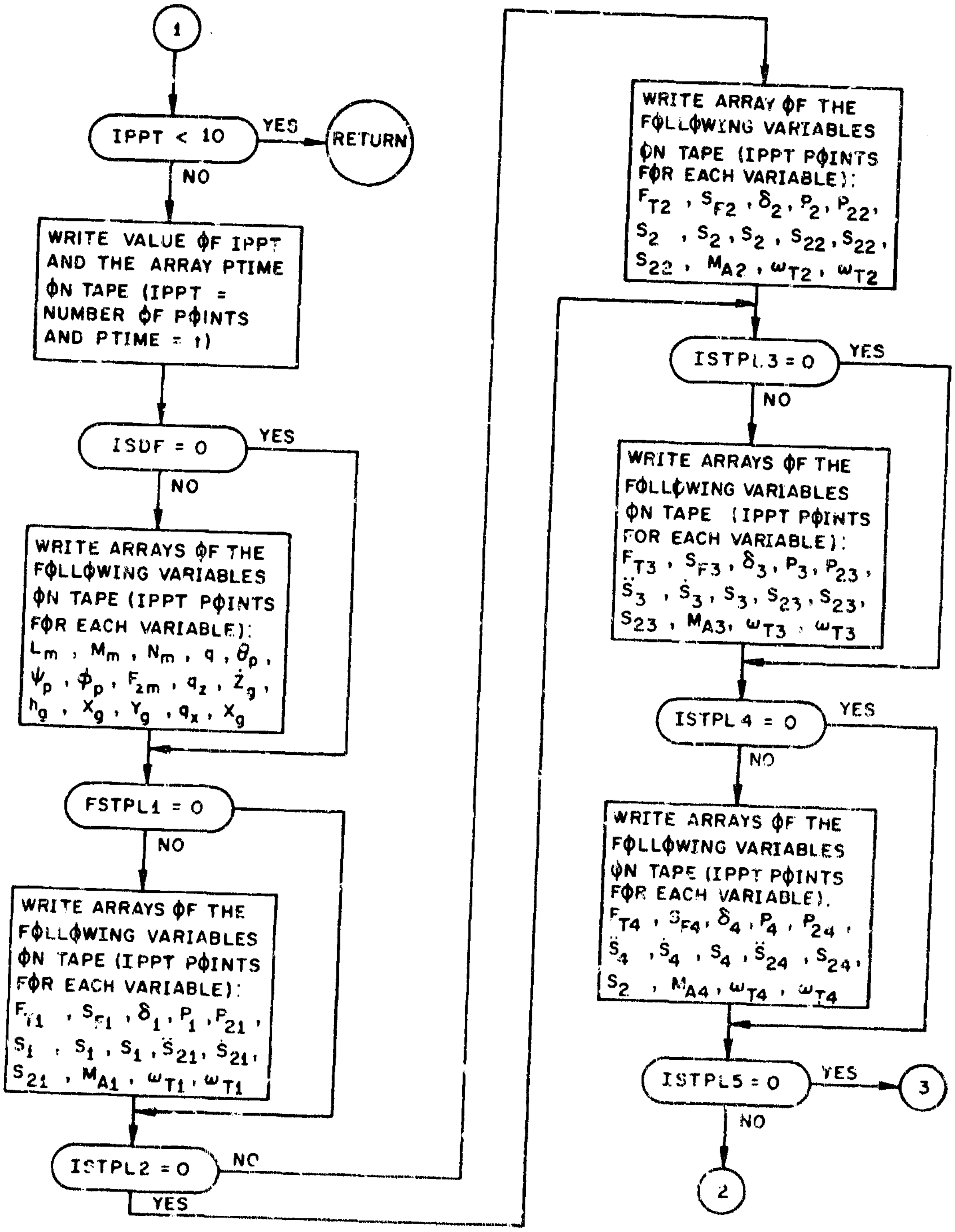
FLOW DIAGRAM, LGEA3C



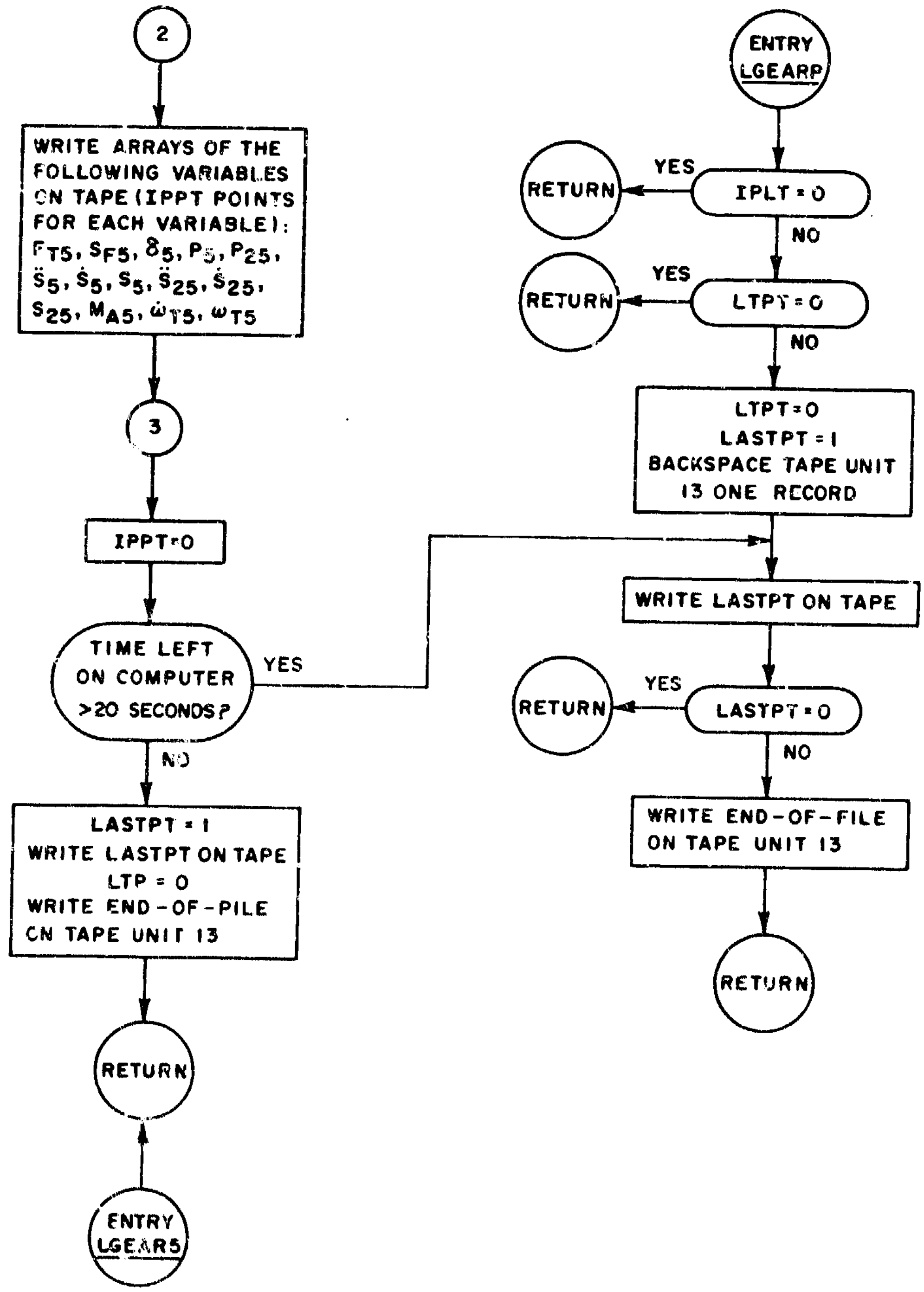
FLOW DIAGRAM (SDFLGP)



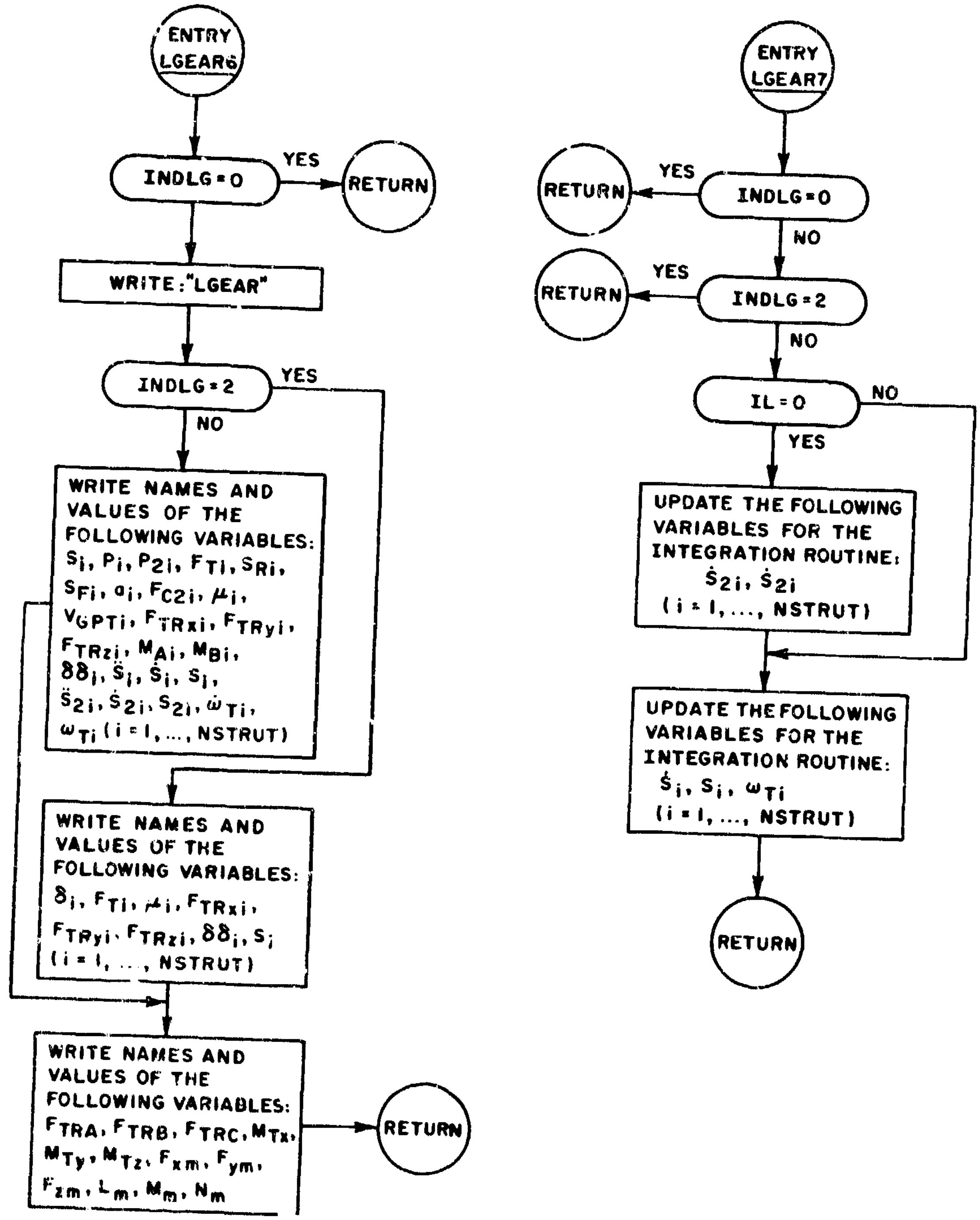
FLOW DIAGRAM (SDFLGP)



FLOW DIAGRAM (SDFLGP)



FLOW DIAGRAM (SDFLGP)



SECTION V

FORTRAN EXTENDED

OVERLAY (1, 0)

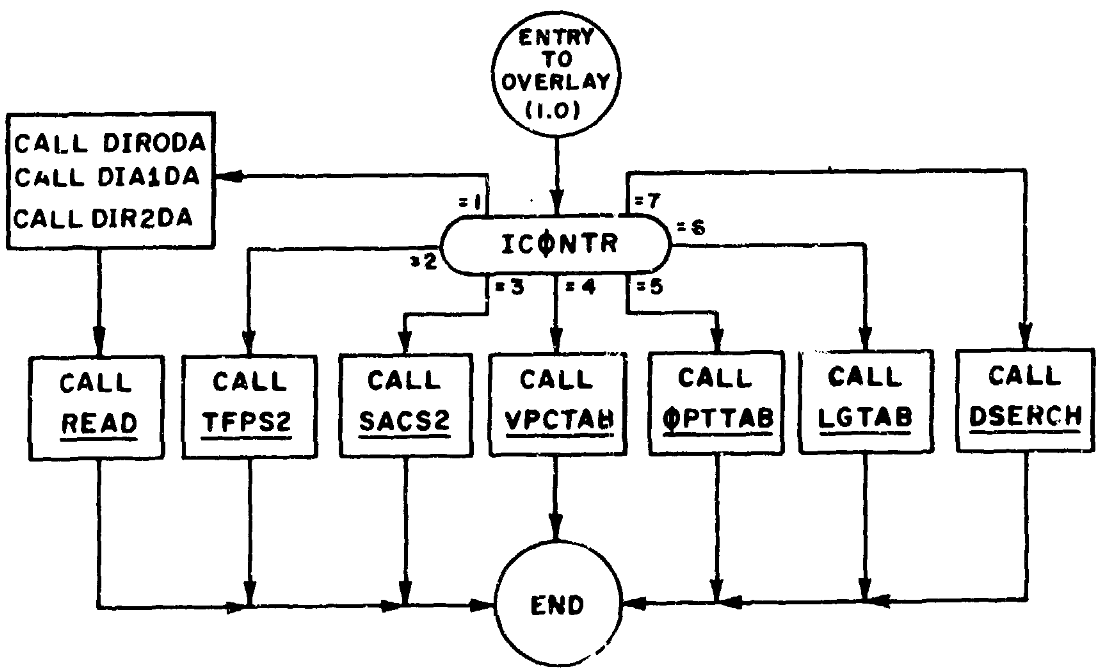
1. TOLAN1 - MAIN PROGRAM FOR OVERLAY

a. Purpose - A main program is required by Fortran Extended for each Overlay.

b. Usage - Linkage to Overlay (1, 0) is provided by the following statement:

CALL OVERLAY (TOLA1, 1, 0)

c. Flow Chart



2. TFFS2 - SET UP TABLE ROUTINE

a. Purpose - To call TSRCH to set up table for various subroutines.

b. Method - When this routine is called by a certain subroutine, it will in turn call TSRCH in order to search the directory for the exact BCD name required by the particular subroutine that called this routine.

c. Usage - Entry is made to the routine with the following statements:

(1) CALL TFFS2

Sets up tables (or provides the table subscripts) for the subroutine TFFS.

(2) CALL SACS2

Sets up tables (or provides the table subscripts) for the subroutine SACS.

(3) CALL VPCTAB

Sets up tables (or provides the table subscripts) for the subroutine VPCS2.

(4) CALL OPTTAB

Sets up tables (or provides the table subscripts) for the subroutine SDF2 (OPT2).

(5) CALL LGTAB

Sets up tables (or provides the table subscripts) for the subroutine LGEAR.

3. READ - INPUT ROUTINE

a. Purpose - To provide a general method of reading a variable field data card and assigning variable length table. Data may be read into symbolic locations in memory.

b. Method - Decimal, Octal, and Integer numbers are converted to binary integers. BCD information is stored in six-character words.

c. Card Formats - The variable name punched in Columns 1-6 is the location into which the first data word is loaded. The variable field information is for relocation. A fixed point integer punched anywhere in the field (67-72) will be treated as a subscript to the variable name punched in Column 1-6. Negative integers punched in columns 67-72 will be in violation of the subroutine. The first blank character found in the card to the right of Column 12 terminates loading from the card. One exception to this is BCD data. Nine 6-character words may be loaded including blanks.

The general character of the data to be loaded is determined by a three letter pseudo-operation punched in columns 8-10. The pseudo-operations are: DEC or blank, ØCT, INT, BCD, and TRA. The pseudo-operation TRA is a method of exit from the subroutine.

d. Decimal Data - Decimal data beginning in column 12 and ending in column 66 is converted to binary and loaded into the symbolic location punched in column 1-6 subscripted by the integer punched in column 67-72. Signs are indicated by + and - preceding the number. All unsigned numbers are treated as positive. If either the character E or . or both appear in the decimal data word, the word is converted to a floating binary number. The decimal exponent used in the conversion is the

number which follows immediately after the character E. This number may have a + or - sign preceding it. If the character E does not appear the exponent is assumed to be zero. If a decimal point does not appear it is assumed to be at the right of the numbers, unless it is the only word or the first word on a card; then it is assumed to be an integer.

All the examples below are equivalent.

(1) 12.345E03

(2) 12.345E+03

(3) 12.345E3

(4) 12345E00

(5) 12345.

(6) 1.2345E4

(7) 1234500E-02

(8) +1234500E-2

Note that in the examples above all decimal words have decimal points. If the frst word on a card, or if it happens to be the only word on a card, does not contain a decimal point, the word will be converted to binary integer.

e. Octal Data-ØCT - The Octal data is loaded the same as decimal data but must have ØCT punched in columns 8, 9, and 10. All data is converted to binary with binary point assumed at the right end of each word.

f. Hollerith Data - BCD - Hollerith information is loaded from columns 3 through 66 and assigned consecutive locations for every 6 characters. A maximum of nine 6-character words may be punched on any one card and the number of words must be punched in column 12. A subscript may also be punched in columns 67-72.

g. Transfer - TRA-- The purpose of the TRA card is to transfer control from the subroutine back to the main program. TRA must be punched in columns 8, 9, and 10. The subscript field is not used. A REWIND may be punched beginning in column 12. Only the R is checked and the only use is for the rewind of a data tape.

h. Integer - INT - Integer data begins in column 12 and ends in column 66. INT is punched in column 8, 9, and 10. It may be relocated with respect to the BCD name by punching a subscript integer in columns 67 through 72. If only one data word is punched per card, columns 8, 9, and 10 may be left blank.

i. Error Messages - A message is written on the output tape describing the type of error encountered. If an error is encountered, execution of the case is deleted and the subroutine only searches for other possible errors in the data. The following error messages are possible.

(1) Symbol not in directory.

(2) Column 12 is blank.

If a bad pseudo-operation is punched in columns 8, 9, and 10 the subroutine will treat it as decimal data.

All checking for redundancies, end of tape, format errors, etc., is handled by FORTRAN system input/output routines.

j. Usage - Linkage to the routine is made by the following statement:

CALL READ

k. Data Preparation - The first card expected by READ is a STCASE TAB with the S beginning in Column 1 and TAB punched in 8, 9, and 10. Following this card is a set of cards which define the table sizes necessary for that case.

Example: CTAB01 10

CTAB02 20

On the above example, CTAB01 is punched beginning in Column 1. The numbers 10 and 20 are punched in Column 12 and indicate the number of machine cells necessary for that table. Any number of tables may be assigned as long as the total number of machine cells does not exceed 600. Follow all table assignments with a TRA punched in Columns 8, 9, and 10. The next data required by the subroutine is a STCASE and followed by any combination of ØCT, BCD, INT, and TRA cards.

4. DSERCH - DIRECTORY SEARCH ROUTINE FOR SUBSCRIPTS

a. Purpose - To provide a method of searching the directory to find the subscript corresponding to a BCD argument.

b. Method - The routine searches the directory for the exact BCD name required by the argument. When an equal compare has been found, the corresponding subscript is returned as a fixed point integer.

c. Usage - Entry is made to the routine with the following statement:

CALL DSERCH (SYM, LØC, IER)

where

SYM = BCD name being search for.

LØC = The location in core

IER = Error Code:

IER is set to 0 if the BCD argument does compare and is not a table name.

IER is set to -1 if the BCD argument does compare and is a table name.

IER is set to +1 if the BCD argument does not compare.

5. TABRE - TABLE DIMENSION SUBSCRIPT ROUTINE

a. Purpose - To compute subscripts such that the table dimension requirements may be variable.

b. Method - Uses input data prepared by the user to compute subscripts for variable table assignments.

c. Usage - Entry is made via the statement

CALL TABRE

When the READ subroutine processes the control card: STCASE TAB then the subroutine TABRE is called by the READ subroutine. STCASE begins in Column 1 and TAB punched in Columns 8, 9, and 10. Following this card will be the cards requesting table sizes, which are read by the TABRE subroutine. As an example the following cards may be read:

TTAB10 10

ATAB01 2

TTAB10 and ATAB01 punched beginning in Column 1 and the required machine cells (10 and 2 in this case) punched beginning in Column 12. Anything punched past Column 15 will not be used. After all table assignments, a TRA should be punched in Columns 8, 9, and 10. The following error messages may be printed:

1. "Symbol does not exist in table list."

2. "Total table sizes exceed N, change maximum total table sizes to NN in subroutines TLU, HIHØ, TFFS, AERØ, and AUXR2," where NN is the required and N is the maximum.

6. TSRCH - TABLE SUBSCRIPT SEARCH ROUTINE

a Purpose - To provide a method of searching the directory for table subscripts.

b. Method - The routine searches the directory for the exact BCD name required by the argument. When an equal has been found the corresponding subscript is returned as a fixed point integer.

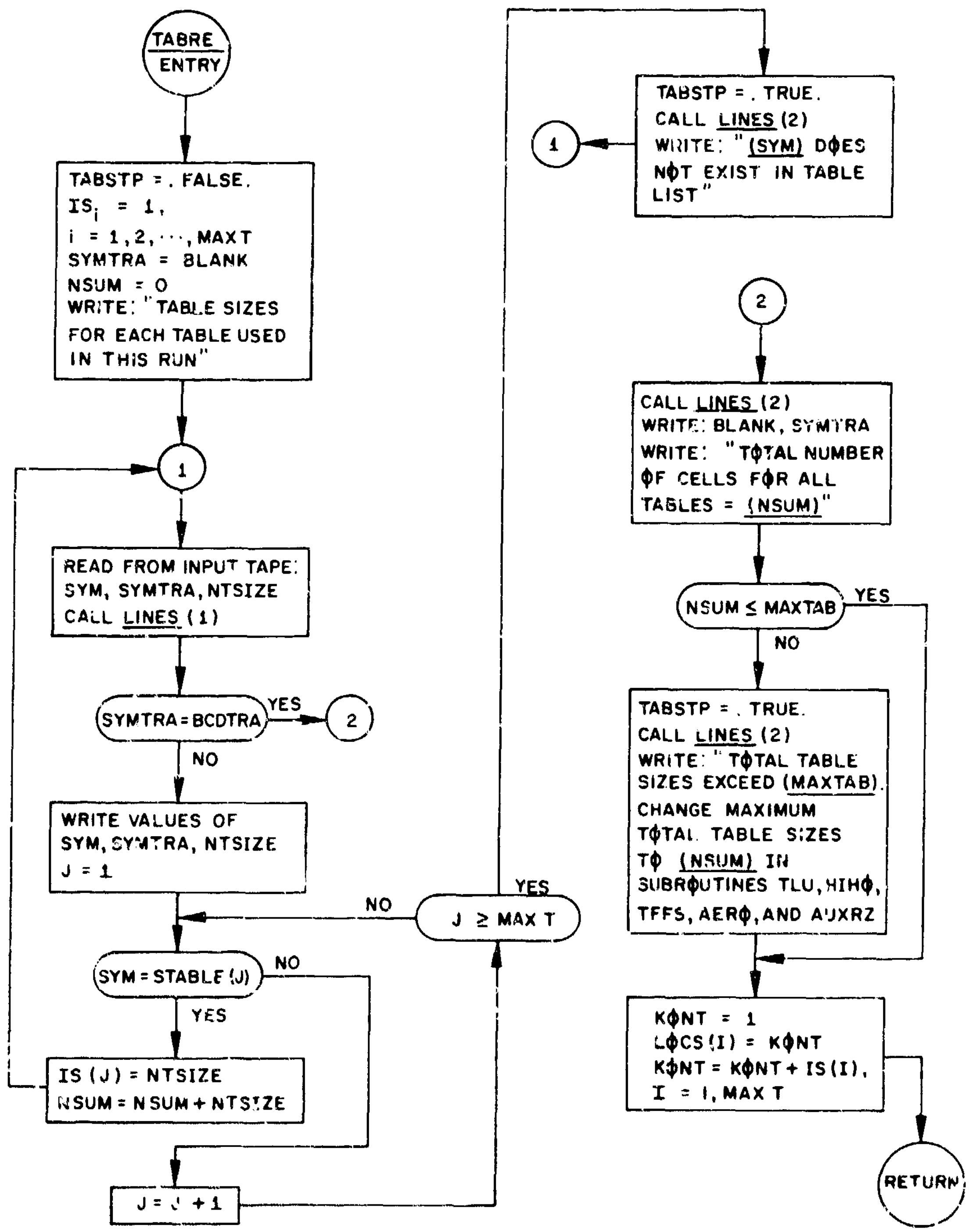
c. Usage - Entry is made to the routine with the following statement:

```
CALL TSRCH (SYM2, LØC2, N2, IER)
```

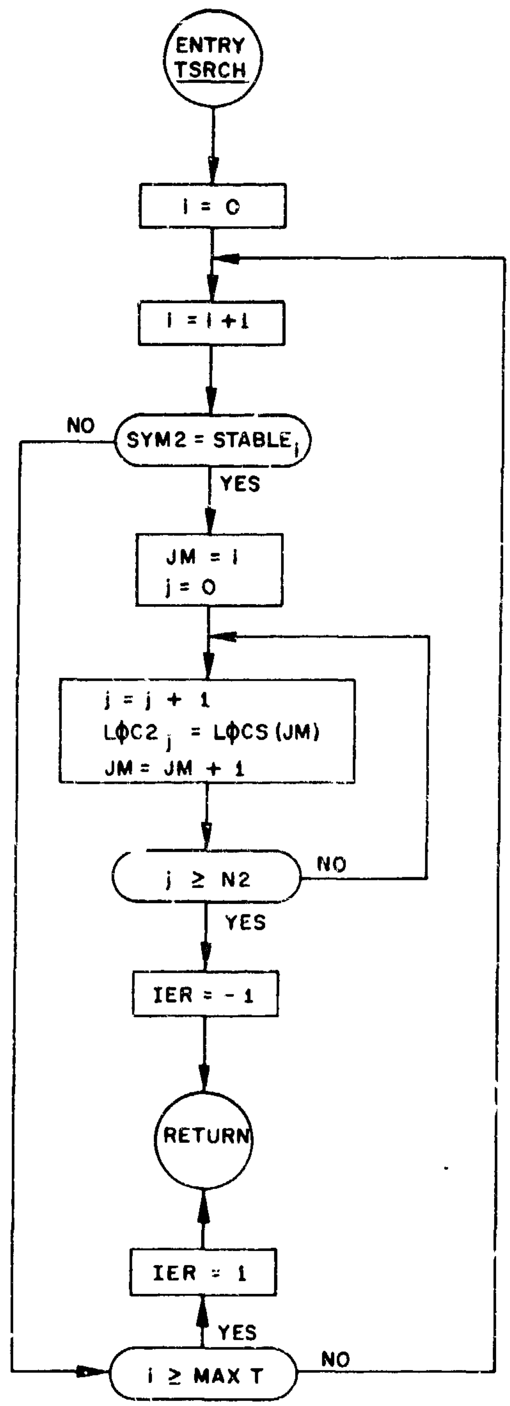
where

SYM2 = BCD name of argument
LOC2 = Location of first subscript
N2 = Number of sequential subscripts to return with
IER = Error Code:
IER is set to +1 if the BCD argument does not compare
IER is set to -1 if the BCD argument does compare.

FLOW DIAGRAM (TABRE.)



FLOW DIAGRAM (TSRCH.)



7. ROUTINES CALLED BY READ SUBROUTINES

The following are subroutines called by the READ subroutine for which flow charts are not provided.

NAME OF SUBROUTINES	LINKAGE
DIPLAC	CALL DIPLAC (RA1, INC, BLANK)
READA	CALL READA (IBCRW)
STORE	CALL STORE (N, INC, INX, STAPE)
WRCARD	CALL WRCARD (MSG)
PACKRR	CALL PACKRR (I1, I2, NNN)
RITE	CALL RITE (IFI, FJ, JJ)

8. DIRODA - INPUT DIRECTORY, PART I

A routine of all the variables in data statements that may be read by the input routine and their corresponding subscript location in COMMON/DIRCOM.

9. DIR1DA - INPUT DIRECTORY, PART II

Continuation of DIRO.

10. DIR2DA - INPUT DIRECTORY, PART III

Continuation of DIR1.

SECTION VI

FORTRAN EXTENDED

OVERLAY (2, 0)

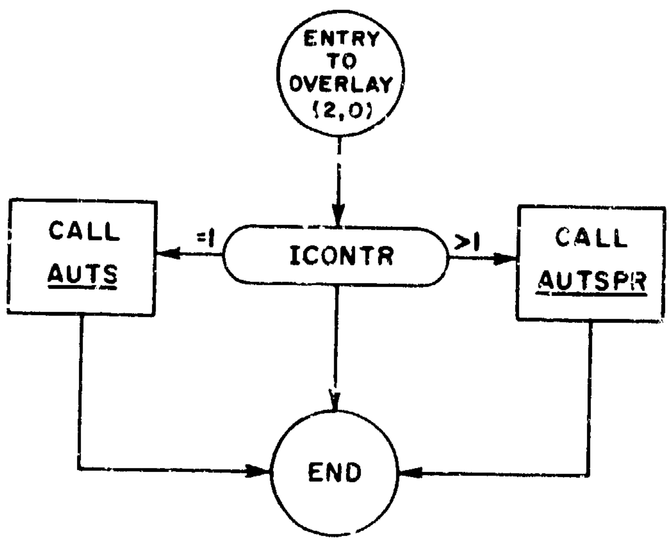
1. TOLAN2 - MAIN PROGRAM FOR OVERLAY

a. Purpose - A main program is required by Fortran Extended for each Overlay.

b. Usage - Linkage to overlay (2, 0) is provided by the following statement:

CALL OVERLAY (TOLA, 2, 0)

c. Flow Chart



2. AUTS - AUTOPILOT AND CONTROL SYSTEM

a. Purpose - The AUTS subprogram calculates the control responses that are needed by the aerodynamic forces and moments subprogram.

b. Usage - Linkage to AUTS is provided by the following statements:

CALL AUTS - Compute the control responses

CALL AUTSPR - Compute the control response and print information as indicated by input parameters.

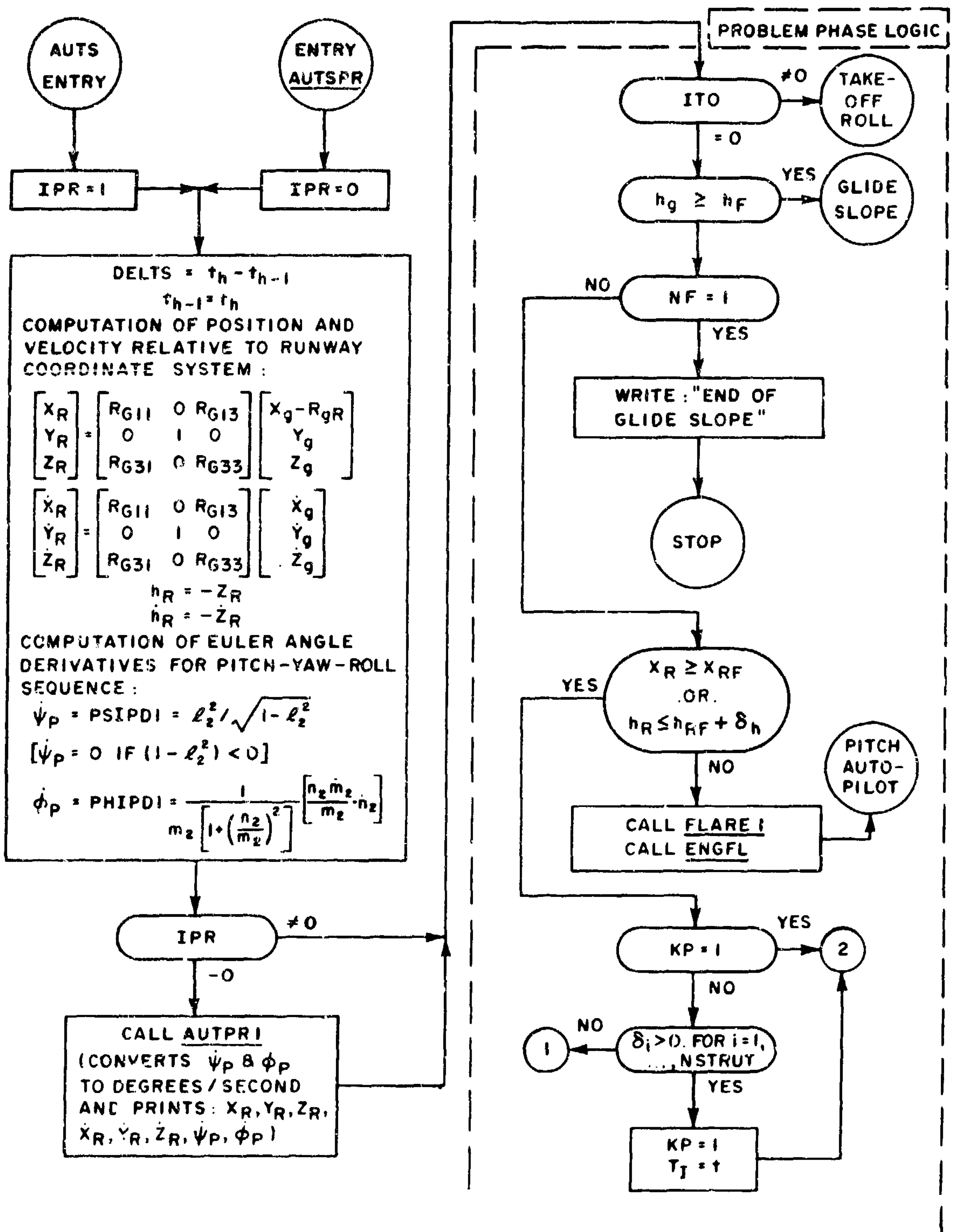
3. FLARE1 - AUTOPILOT FLARE

a. Purpose - The FLARE subroutine calculates the actual selected touchdown conditions, time to touchdown for X and Y constraints, required accelerations to meet touchdown conditions, airspeed flight path angle relative to runway, desired thrust, and angle of attack.

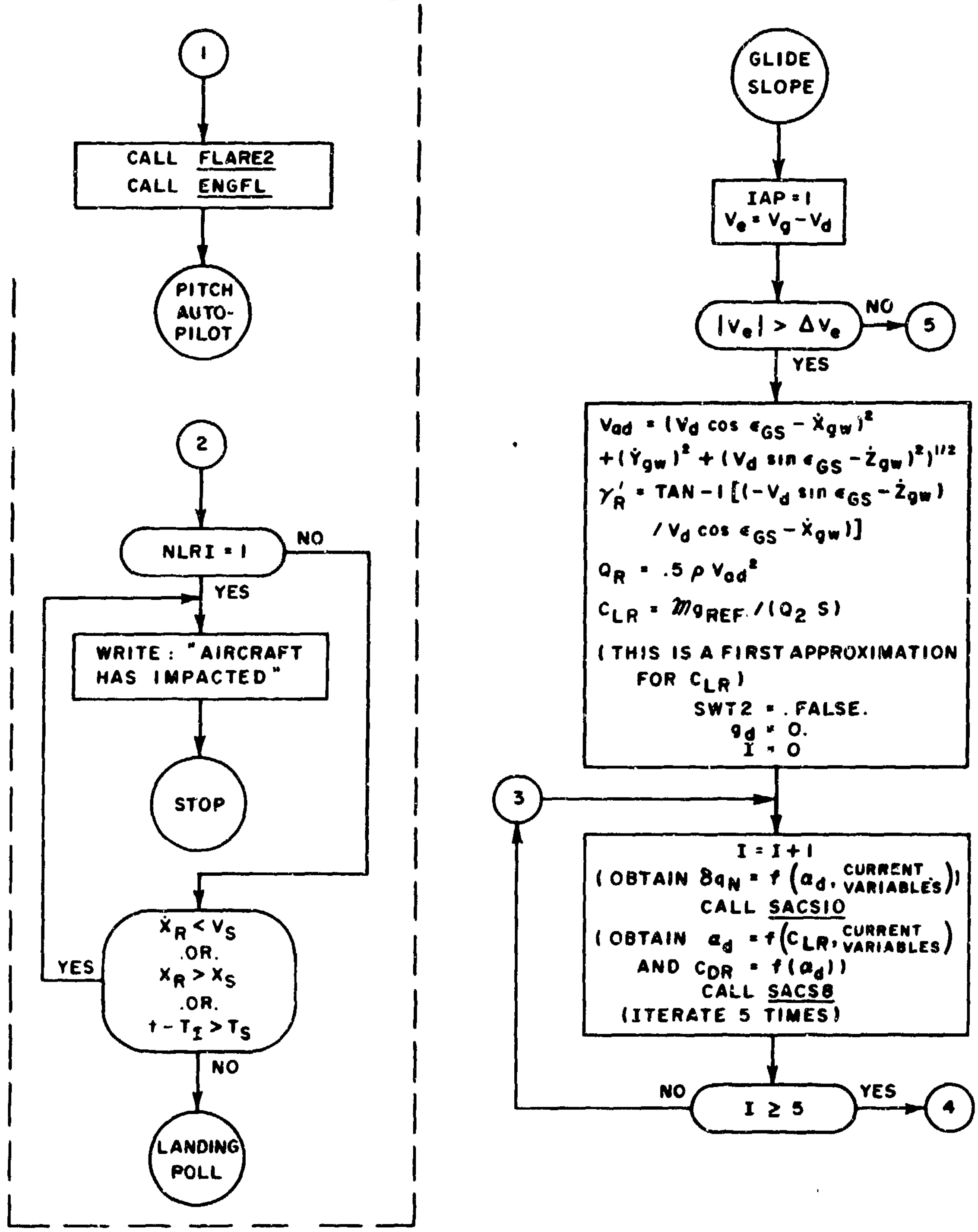
b. Usage - Linkage to FLARE is provided by the following statements:

CALL FLARE1 (IPR)	Computes all that is mentioned above. IPR is the print indicator. If IPR = 0, the print routine is called.
CALL FLARE2 (IPR)	Only computes airspeed flight path angle relative to runway, desired Euler roll angle and desired angle of attack.

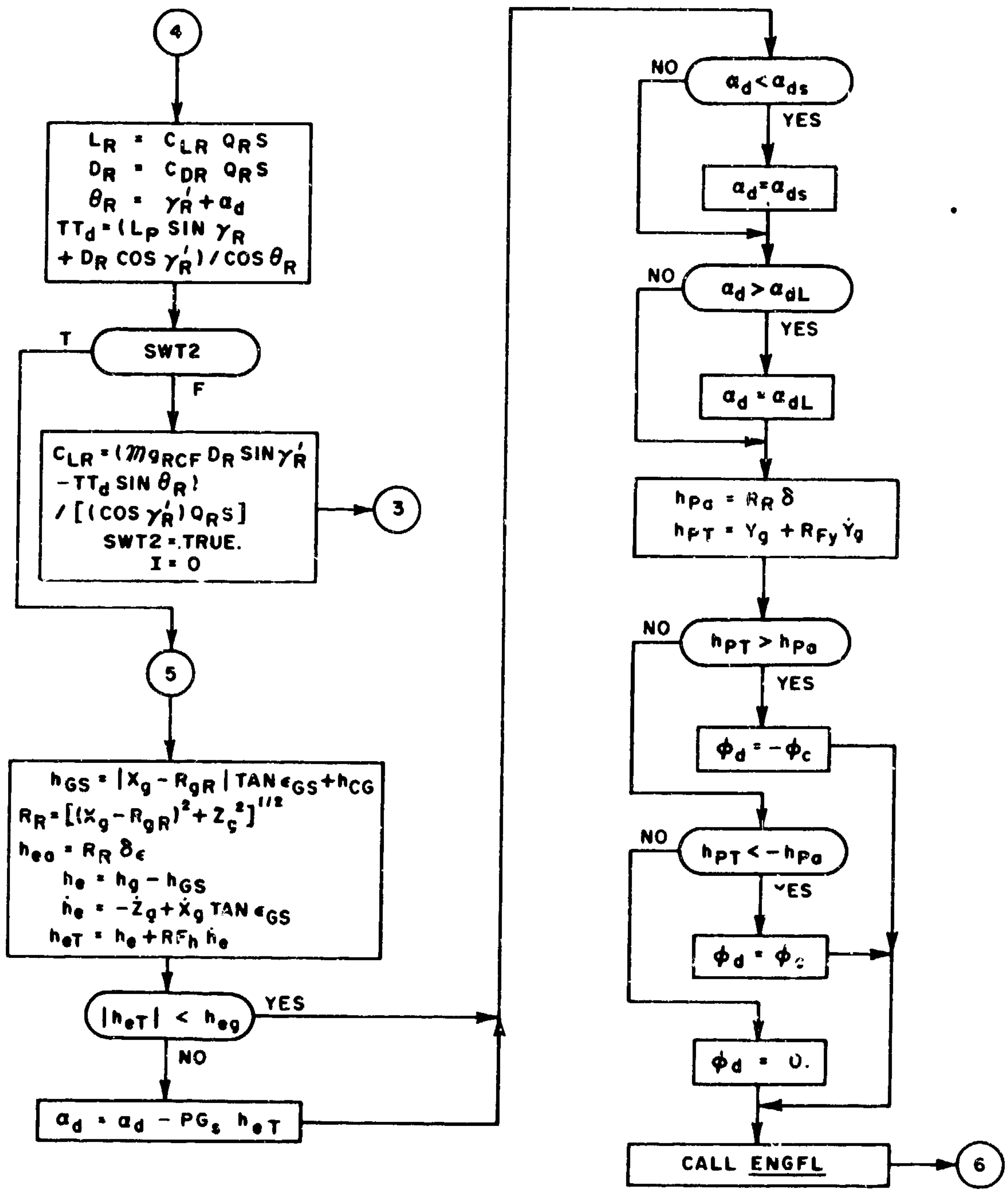
FLOW DIAGRAM (AUTS)



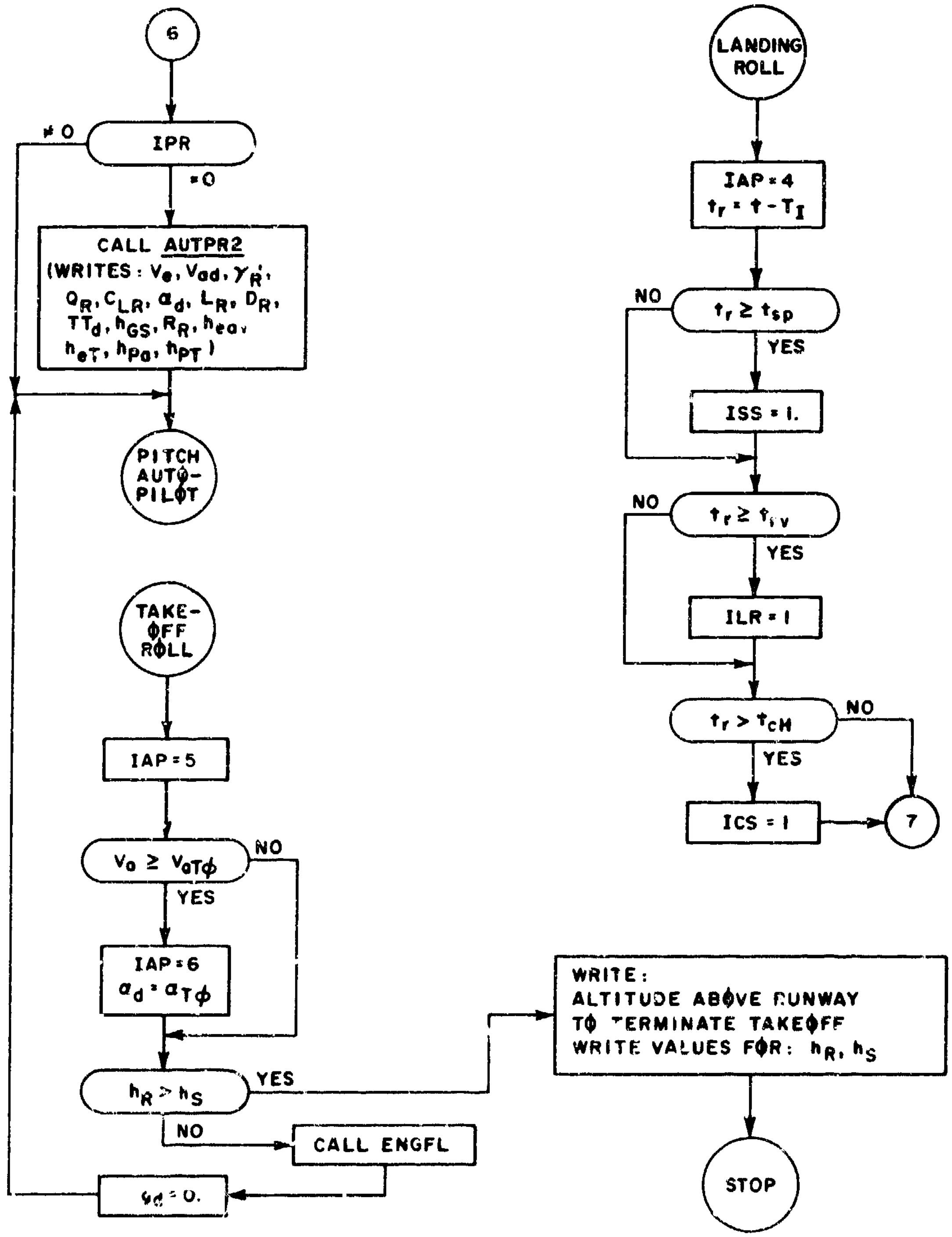
FLOW DIAGRAM (AUTS)



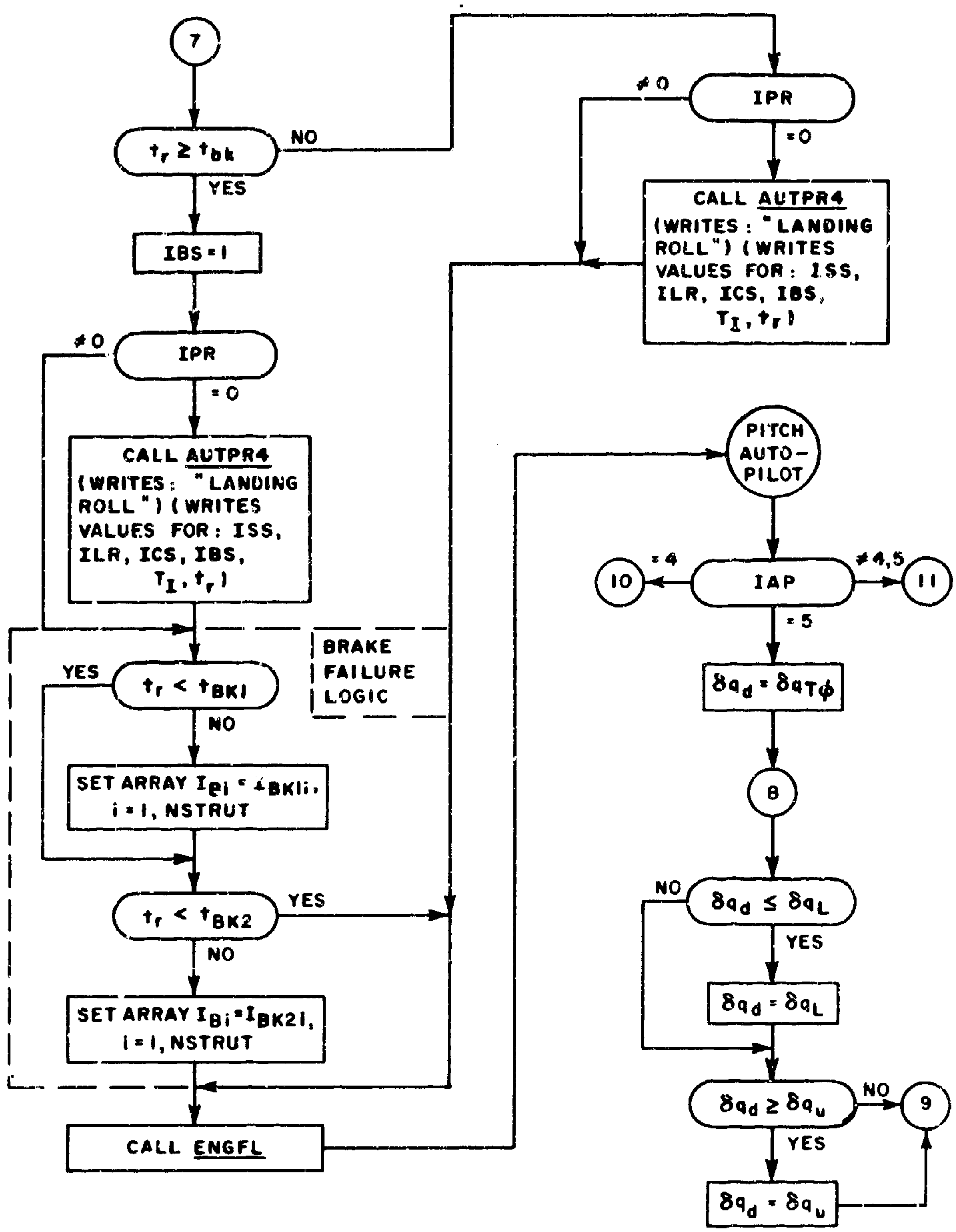
FLOW DIAGRAM (AUTS)



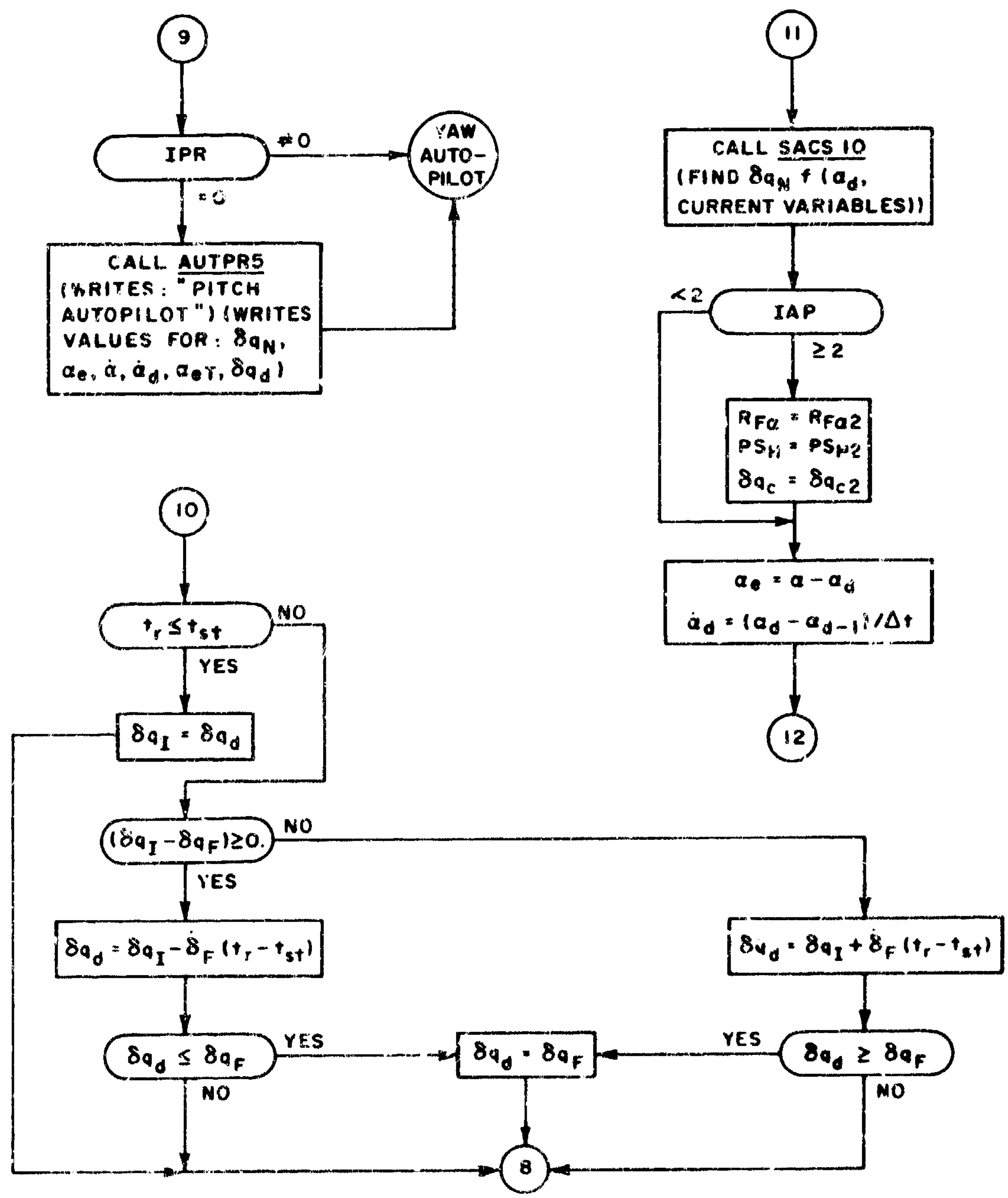
FLOW DIAGRAM (AUTS)



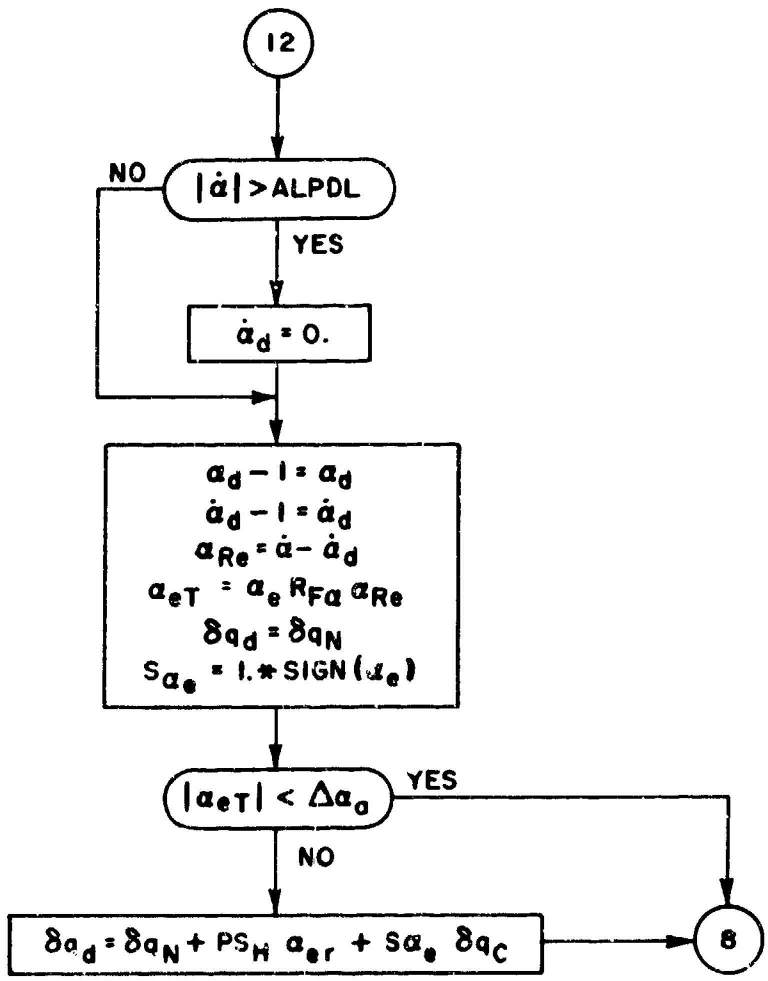
FLOW DIAGRAM (AUTS.)



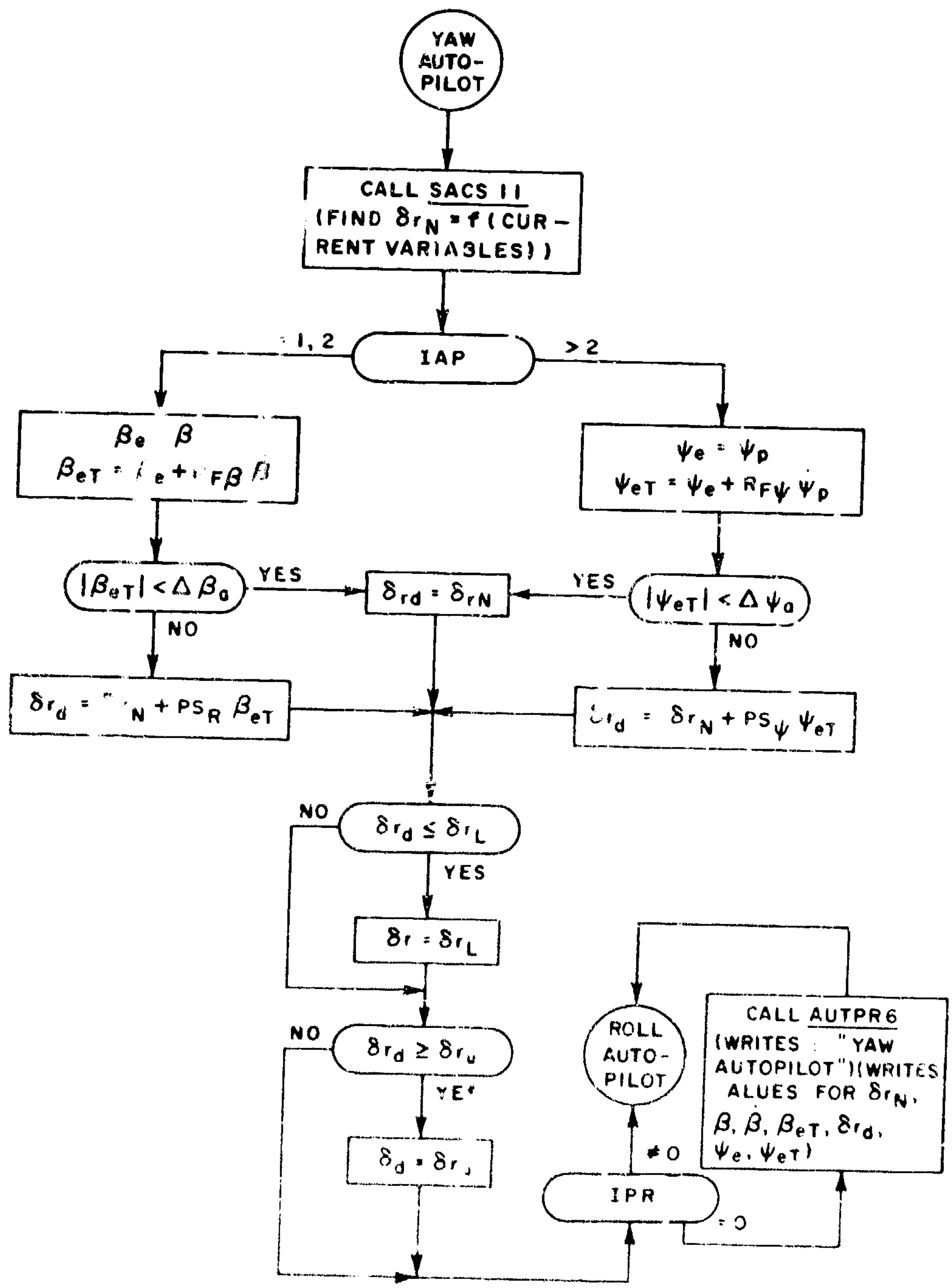
FLOW DIAGRAM (AUTS)



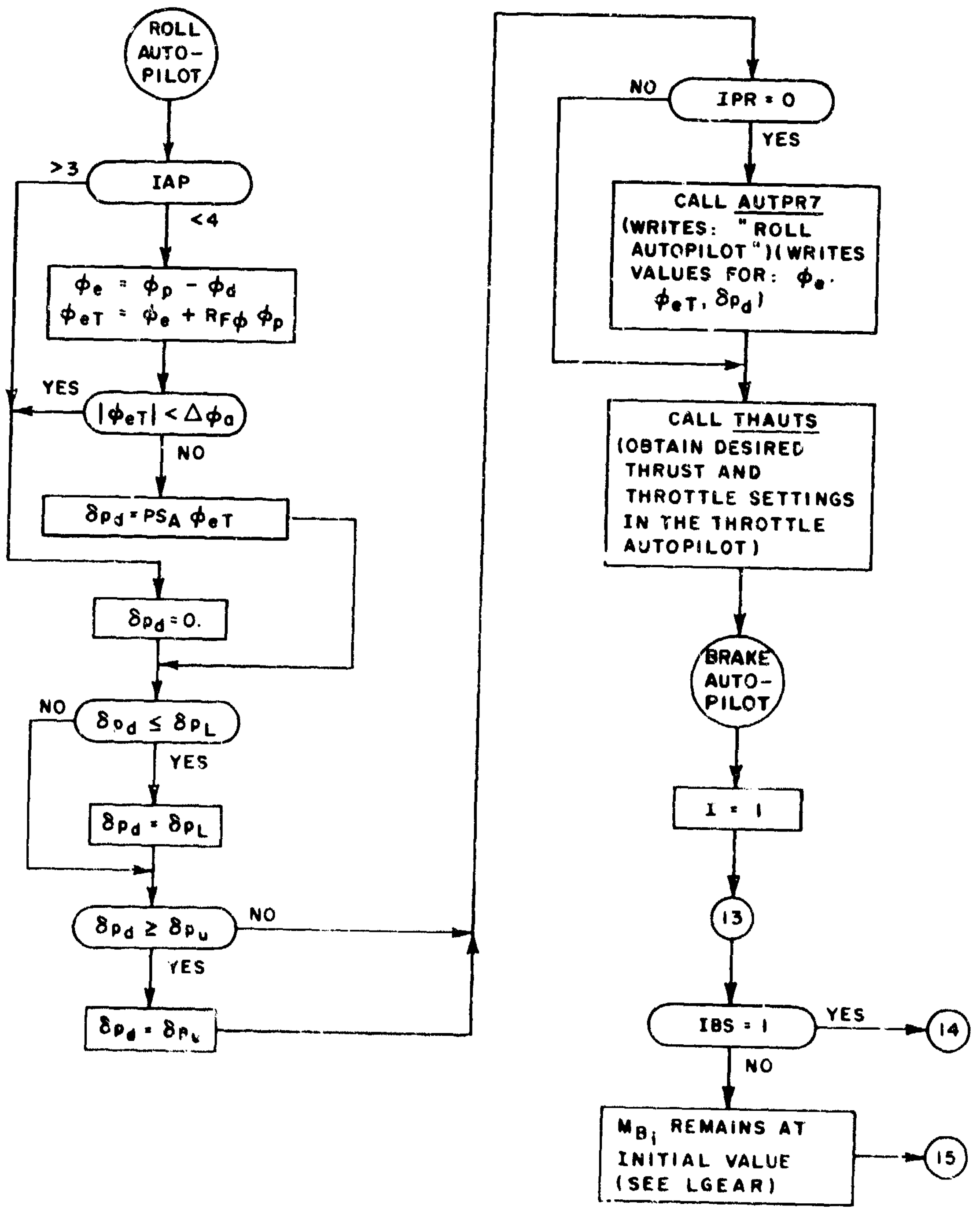
FLOW DIAGRAM (AUTS.)



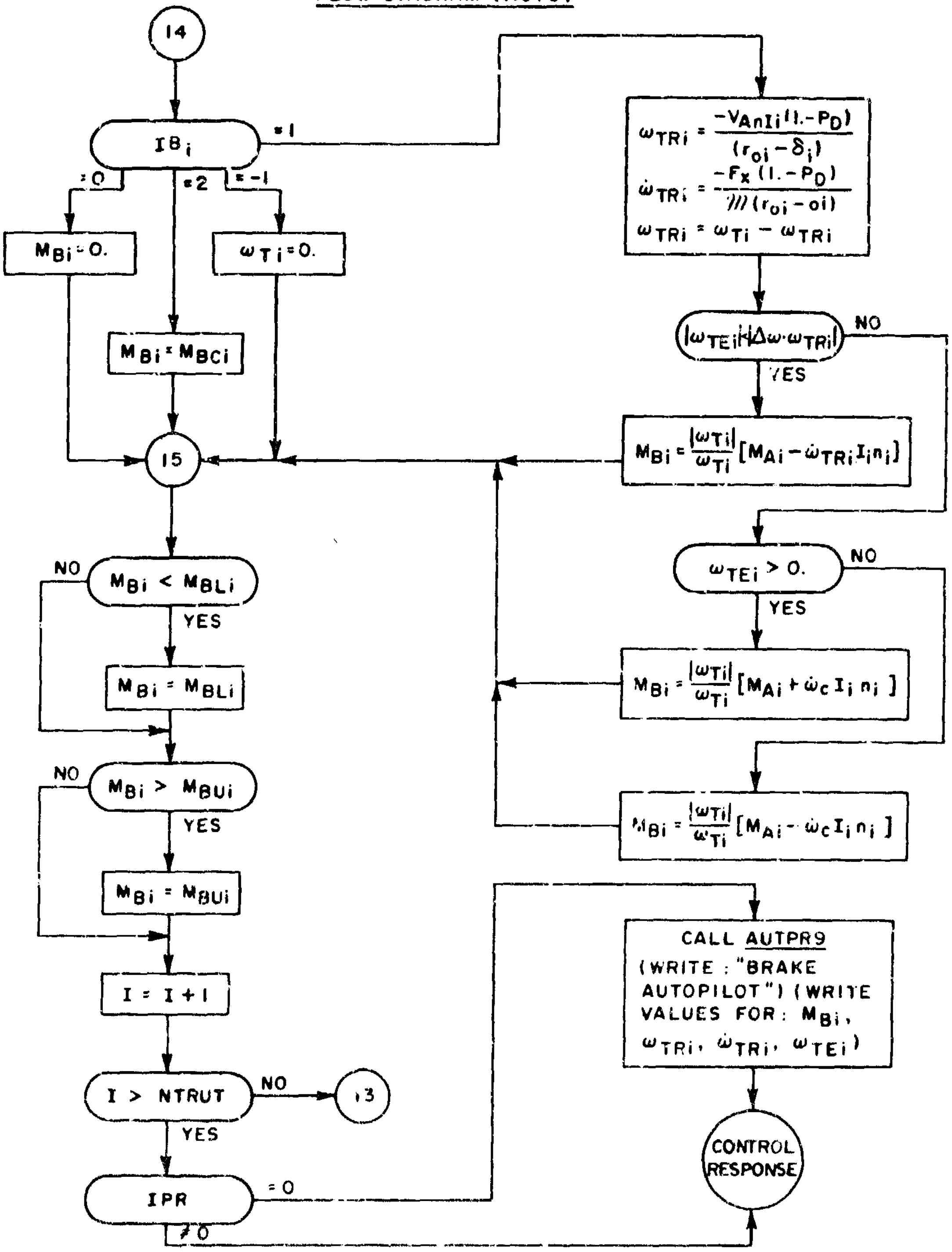
FLOW DIAGRAM (AUTS)



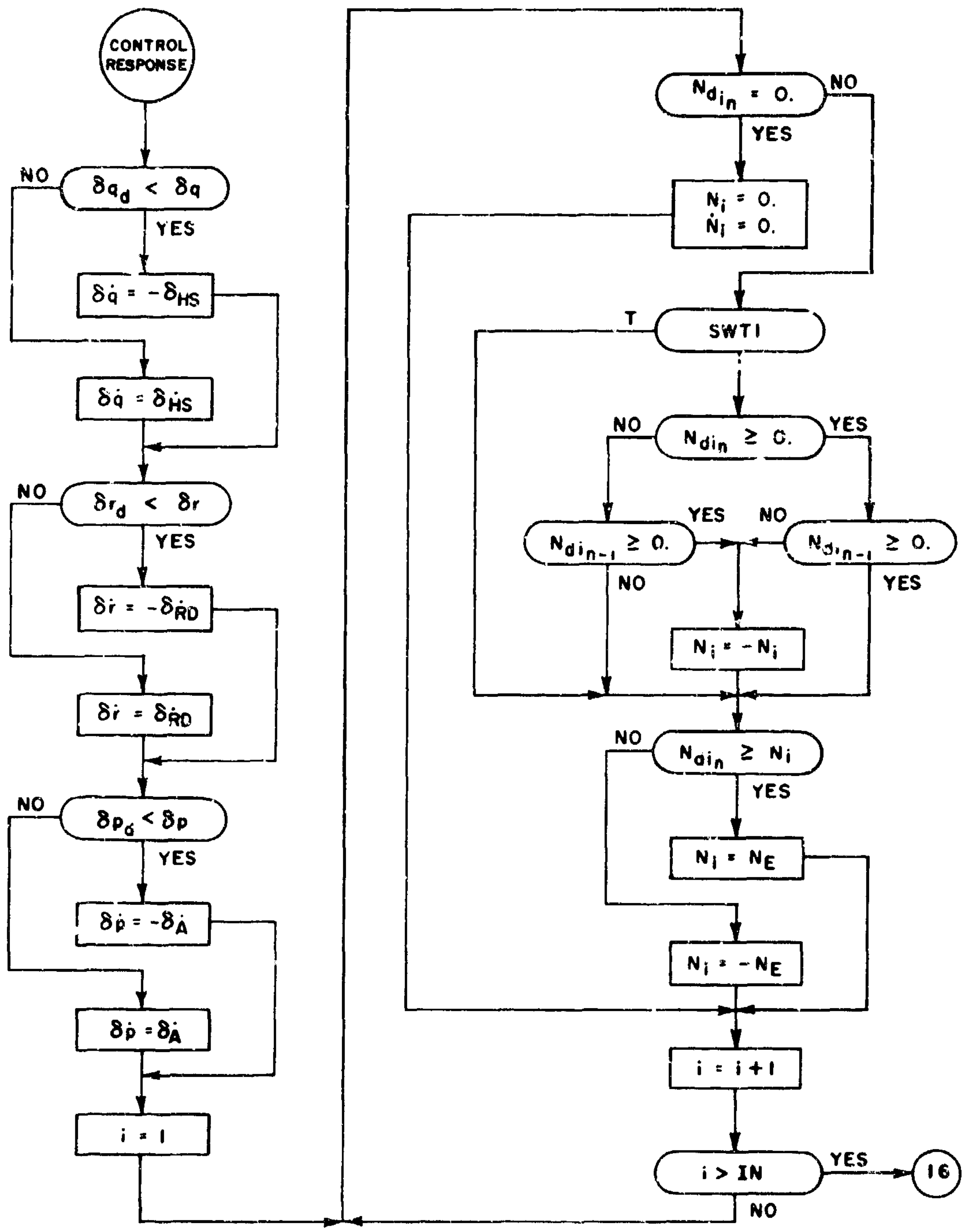
FLOW DIAGRAM (AUTS.)



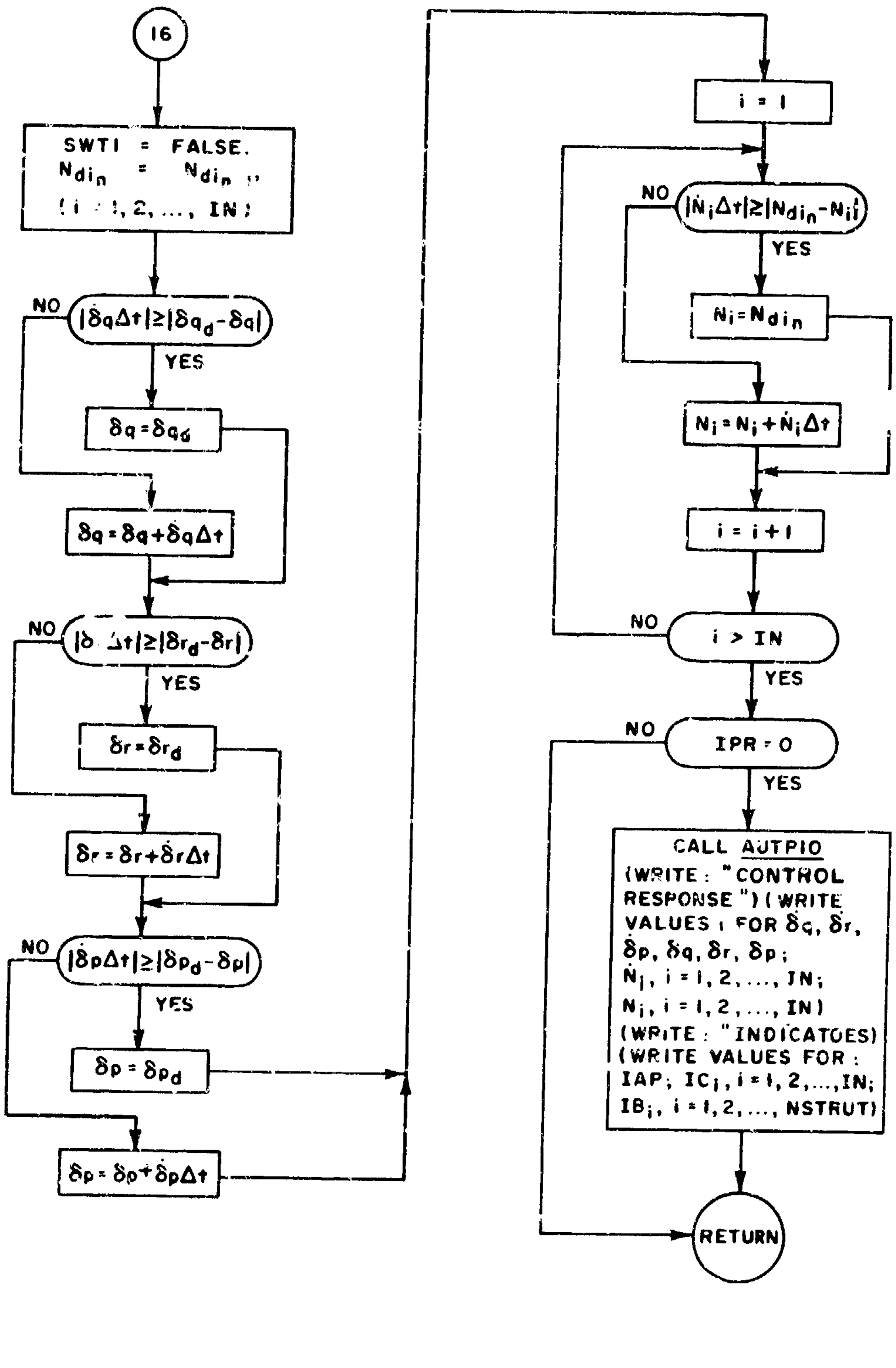
FLOW DIAGRAM (AUTS)



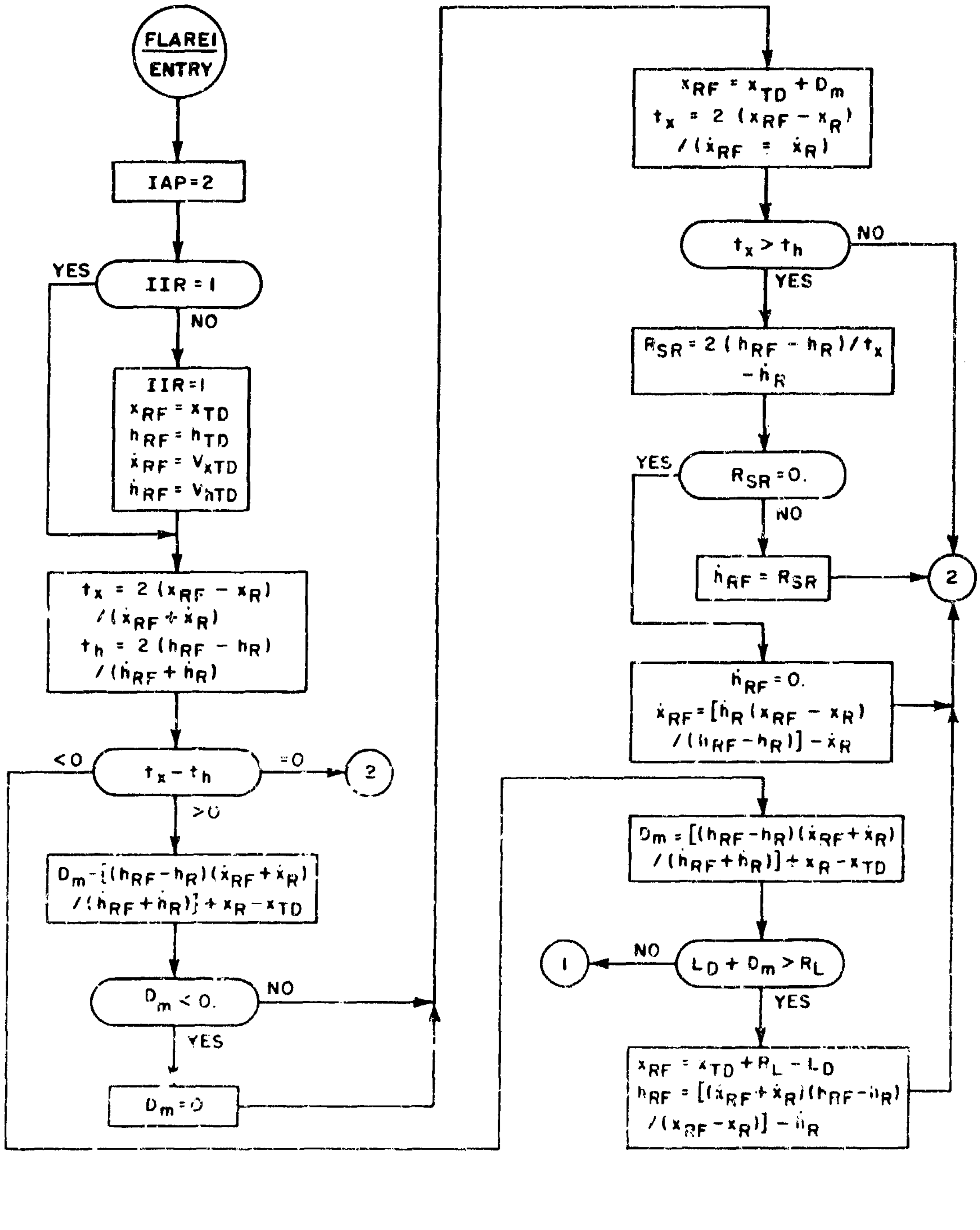
FLOW DIAGRAM (AUTS.)



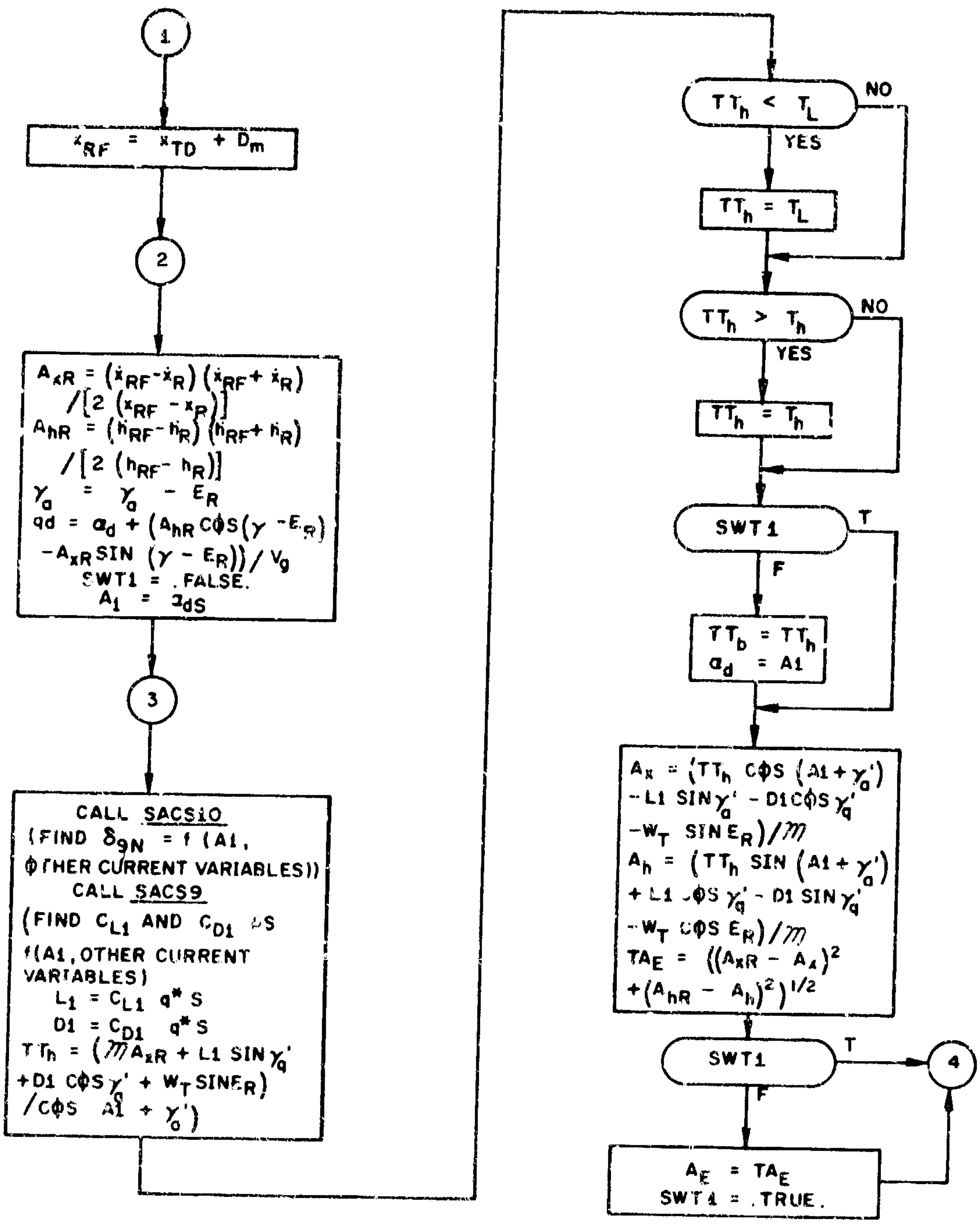
FLOW DIAGRAM (AUTS.)



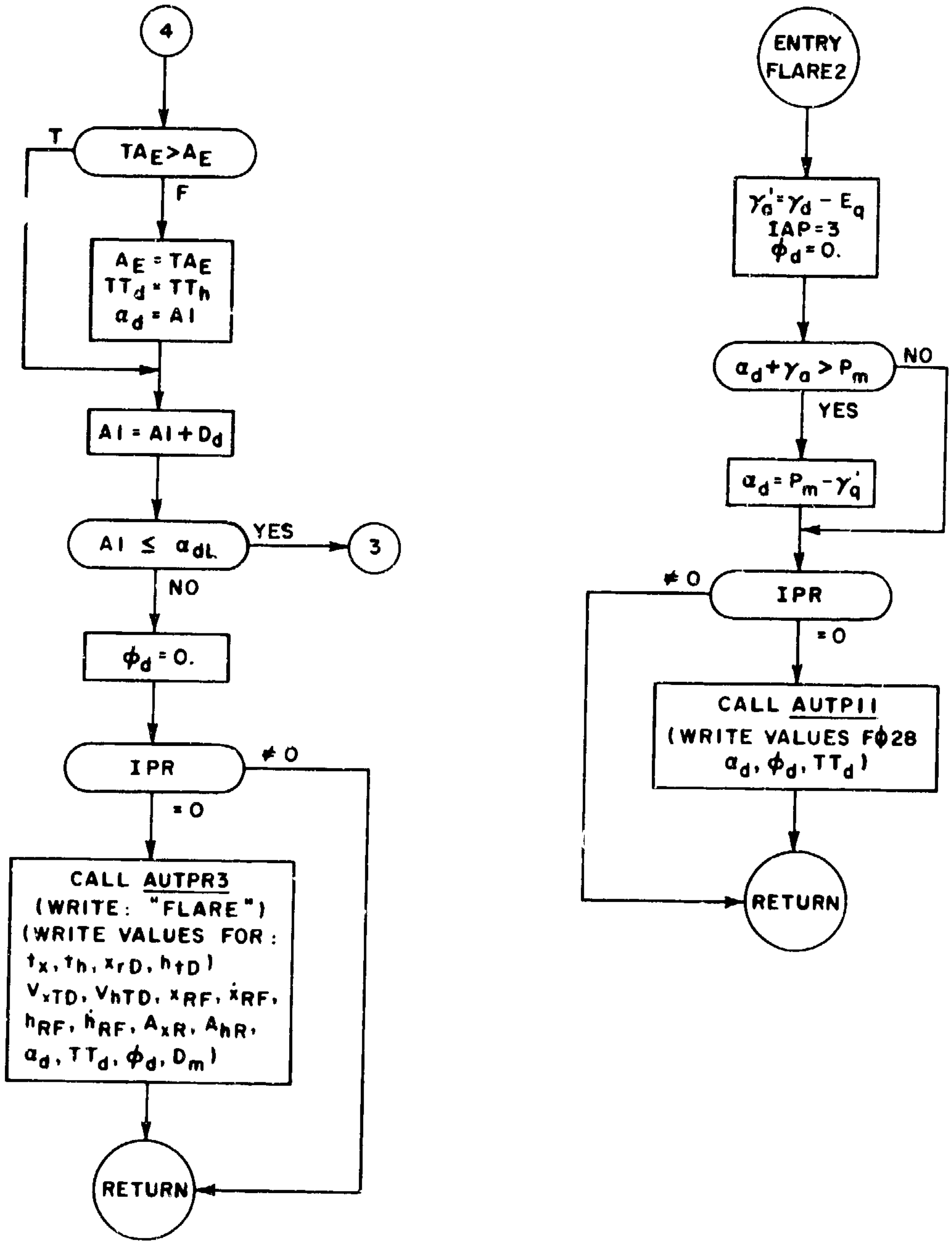
FLOW DIAGRAM (FLARE I)



FLOW DIAGRAM (FLARE 1)



FLOW DIAGRAM (FLARE I)



4. AUTPR1 - AUTOPILOT PRINT ROUTINE

a. Purpose - The AUTPR subroutine prints output for the autopilot.

b. Usage - Linkage to AUTPR is provided by the following statements:

(1) CALL AUTPR1 — Write "AUTS".
If AUXICP ≠ 0, WRITE
<u>"AUXILIARY COMPUTATIONS"</u>
and the following variables: X_R, Y_R, Z_R, $\dot{X}_R$, $\dot{Y}_R$, $\dot{Z}_R$, $\dot{\psi}_p$, $\dot{\phi}_p$

(2) CALL AUTPR2 — If MANLØG ≠ 0, write <u>"GLIDE SLOPE"</u> and the following variables: Ve, V_{ad}, γ_R^1, Q_R, C_{L_R}, α_d, L_R, D_R, TTd, H_{GS}, R_R, h_{ea}, h_e, h_{eT}, h_{Pa}, h_{PT}, ϕ_d

(3) CALL AUTPR3 — If MANLØG ≠ 0, write <u>"FLARE"</u>, and the following variables: t_x, t_h, X_{TD}, h_{TD}, Vx_{TD}, Vh_{TD}, X_{RF}, $\dot{X}_{RF}$, h_{RF}, $\dot{h}_{RF}$, A_{XR}, A_{hR}, α_d, T_{Td}, ϕ_d

(4) CALL AUTPR4 — If MANLØG ≠ 0, write <u>"LANDING ROLL"</u>, and the following variables: ISS, ILR, ICS, IBS, T_I, t_r

(5) CALL AUTPR5 — If PITCHP ≠ 0, write <u>"PITCH AUTO PILOT,"</u> and the following variables: δ_{qN}, α_e, $\dot{\alpha}$, $\dot{\alpha}_d$, α_{eT}, δ_{qd}

(6) CALL AUTPR6 — If YAWAUP ≠ 0, write <u>"YAW AUTØPILØT"</u>, and the following variables; δ_{rN}, β, $\dot{\beta}$, β_{eT}, δ_{rd}, ψ_e, ψ_{et}

(7) CALL AUTPR7 — If RØLLAP ≠ 0, write "ROLL AUTØPILØT"
and the following variables:
ψ_e, ψ_{eT}, δ_{pd}

(8) CALL AUTPR8 — If THRØAP $\neq$ 0, write "THRØTTLE AUTØPILØT", and the following variables N_{di}, T_{d_i}, i = 1, 2, ..., IN

(9) CALL AUTPR9 — If BRAKAP $\neq$ 0, write "BRAKE AUTØPILØT" and the following variables: M_{Bi}, i = 1, 2, ..., NSTRUT. Also if BRAKAP $\neq$ 0 and IBS = 1, write the following brake autopilot variables: ω_{tri}, ω_{TRi}, ω_{TEi}, i = 1, 2, ..., NSTRUT

(10) CALL AUTP10 — If CØNTRP $\neq$ 0, write "CØNTRØL RESPØNSE", and the following variables: δ_q, δ_r, δ_p, δ_q, δ_r, δ_p, $\dot{N}_i$, N_i, i = 1, 2, ..., IN. Also, if INDICP $\neq$ 0, write "INDICATØRS" and the following variables: IAP, IC_i, i = 1, 2, 3, 4; IB_i, i = k, 2, 3, 4, 5.

(11) CALL AUTP11 — If MANLØG $\neq$ 0, write "FLARE", and the following variables: α_d, $\emptyset_d$, T_{Td}

5. THAUTS - AUTOPILOT THROTTLE

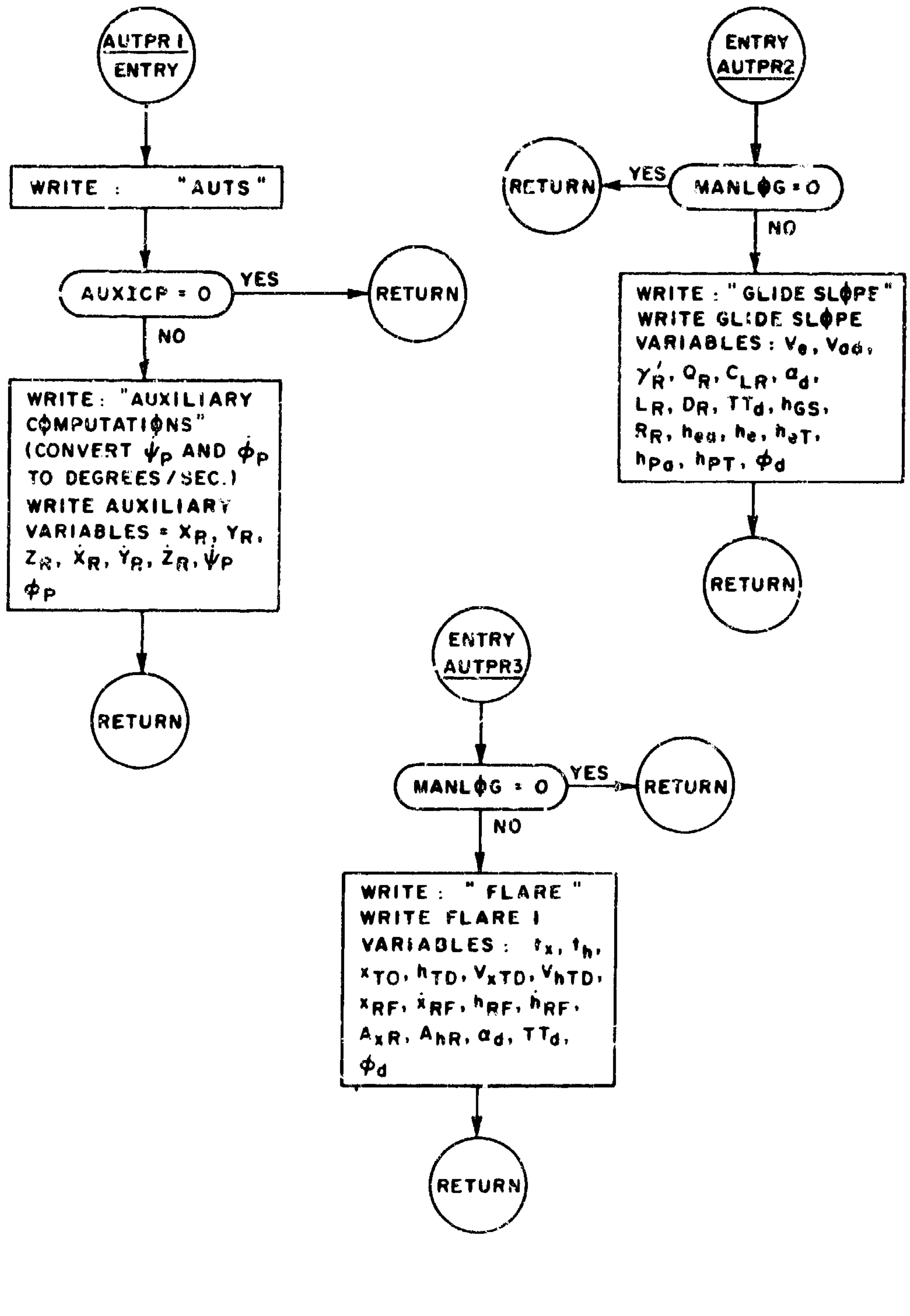
a. Purpose - To compute the desired setting and desired thrust for each engine.

b. Usage - Linkage to the throttle autopilot is made by the following statement.

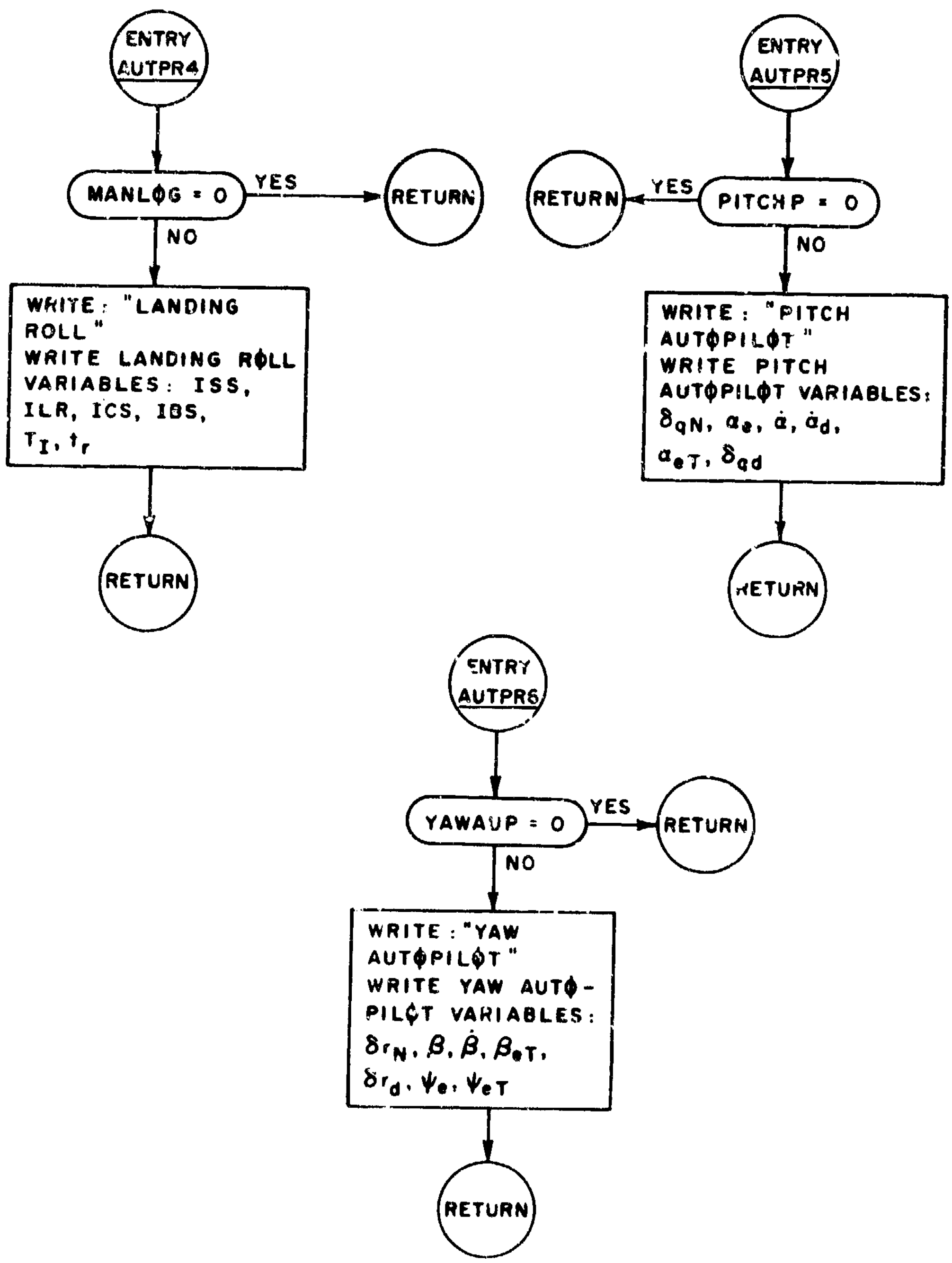
CALL THAUTS (IPR)

If IPR = 0, the print routine is called to print the throttle autopilot output variables.

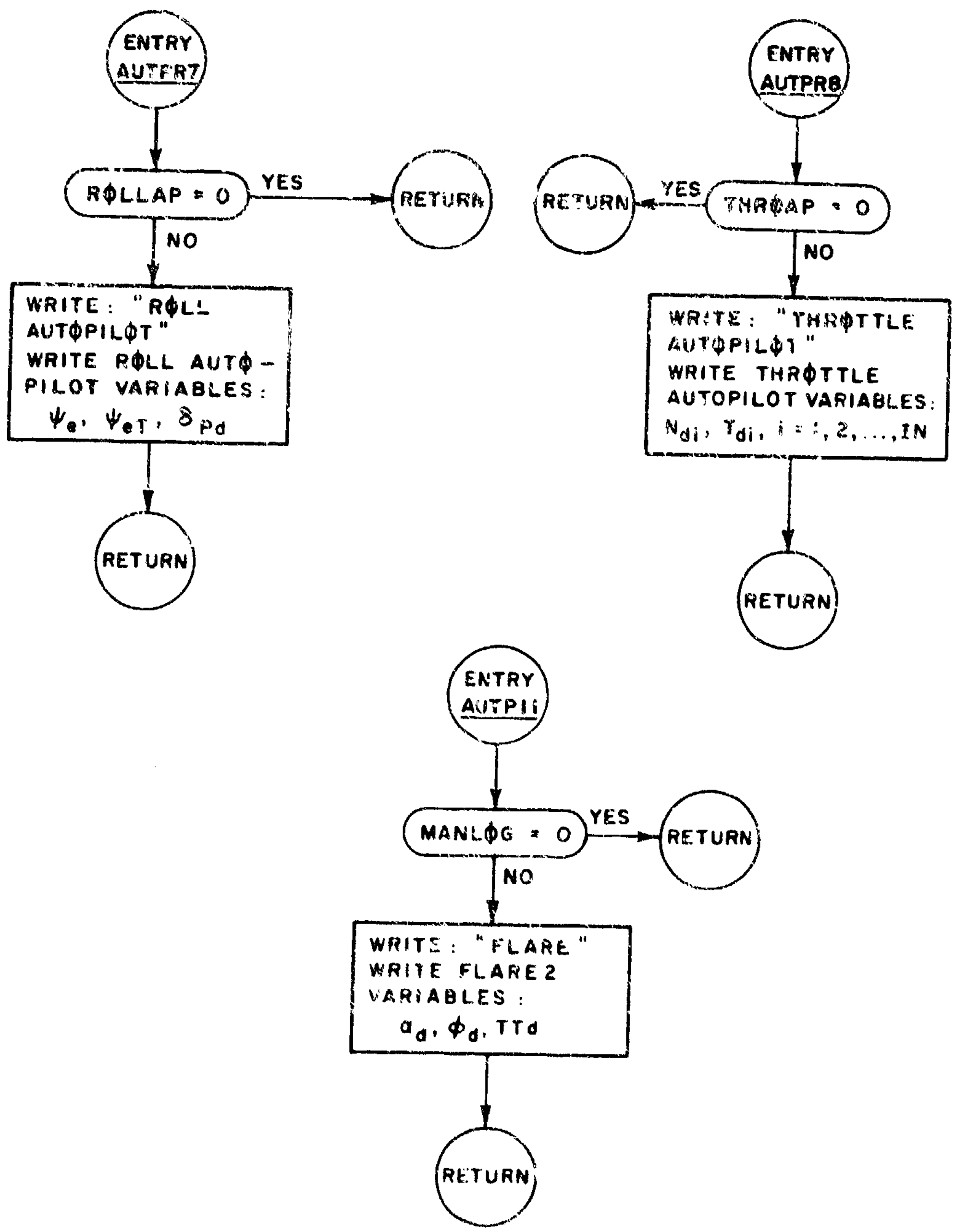
FLOW DIAGRAM (AUTPR1)



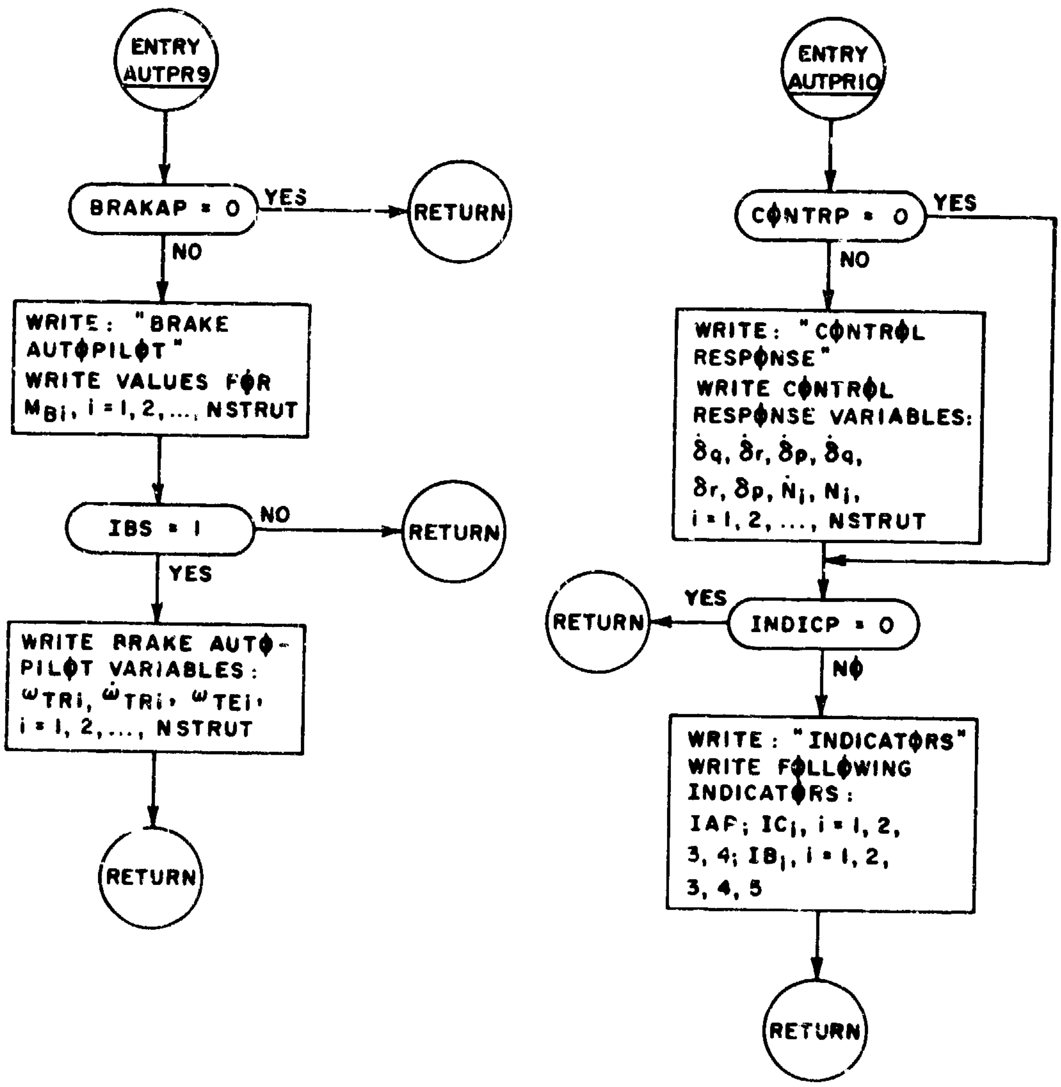
FLOW DIAGRAM (AUTPRI)



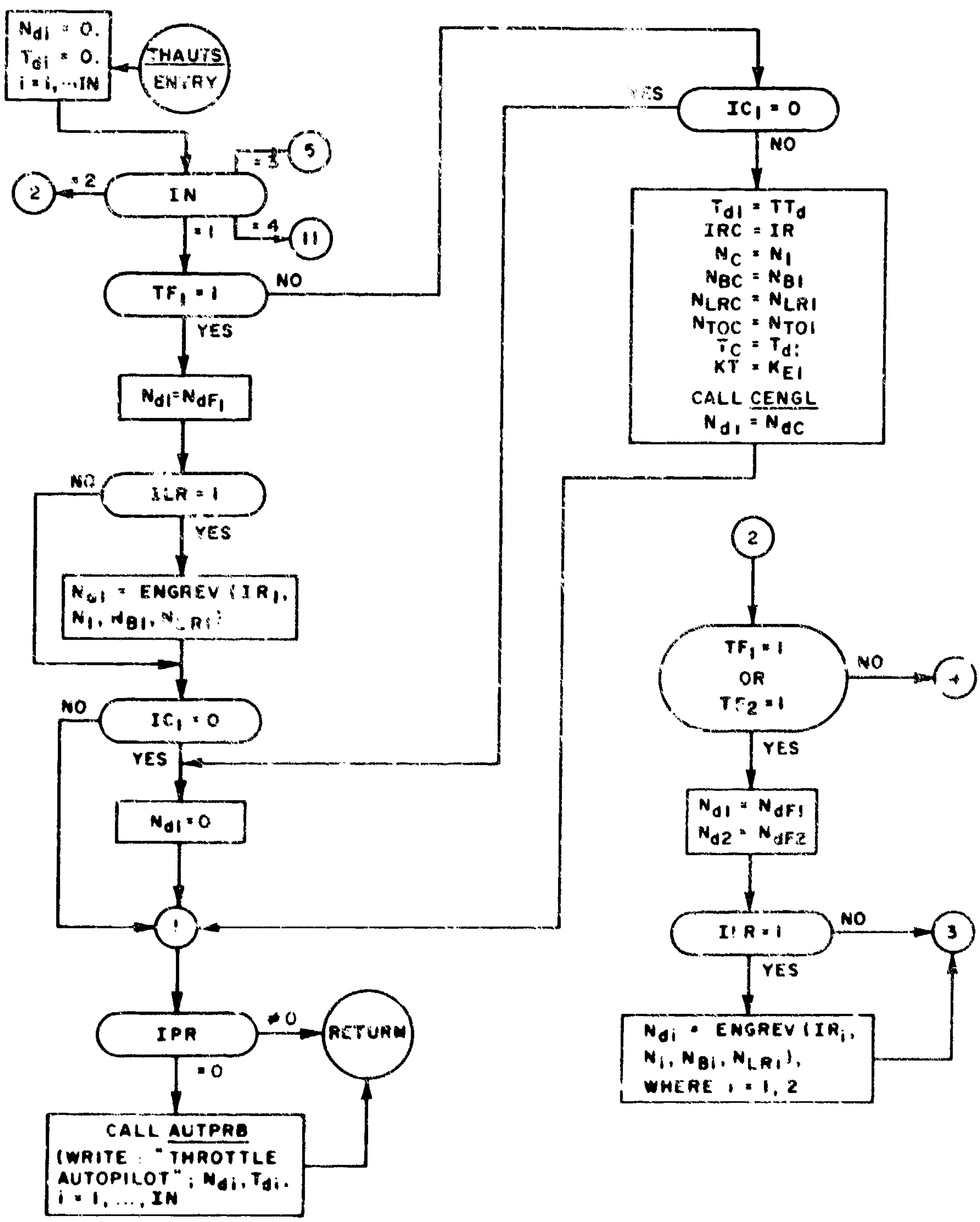
FLOW DIAGRAM (AUTPRI)



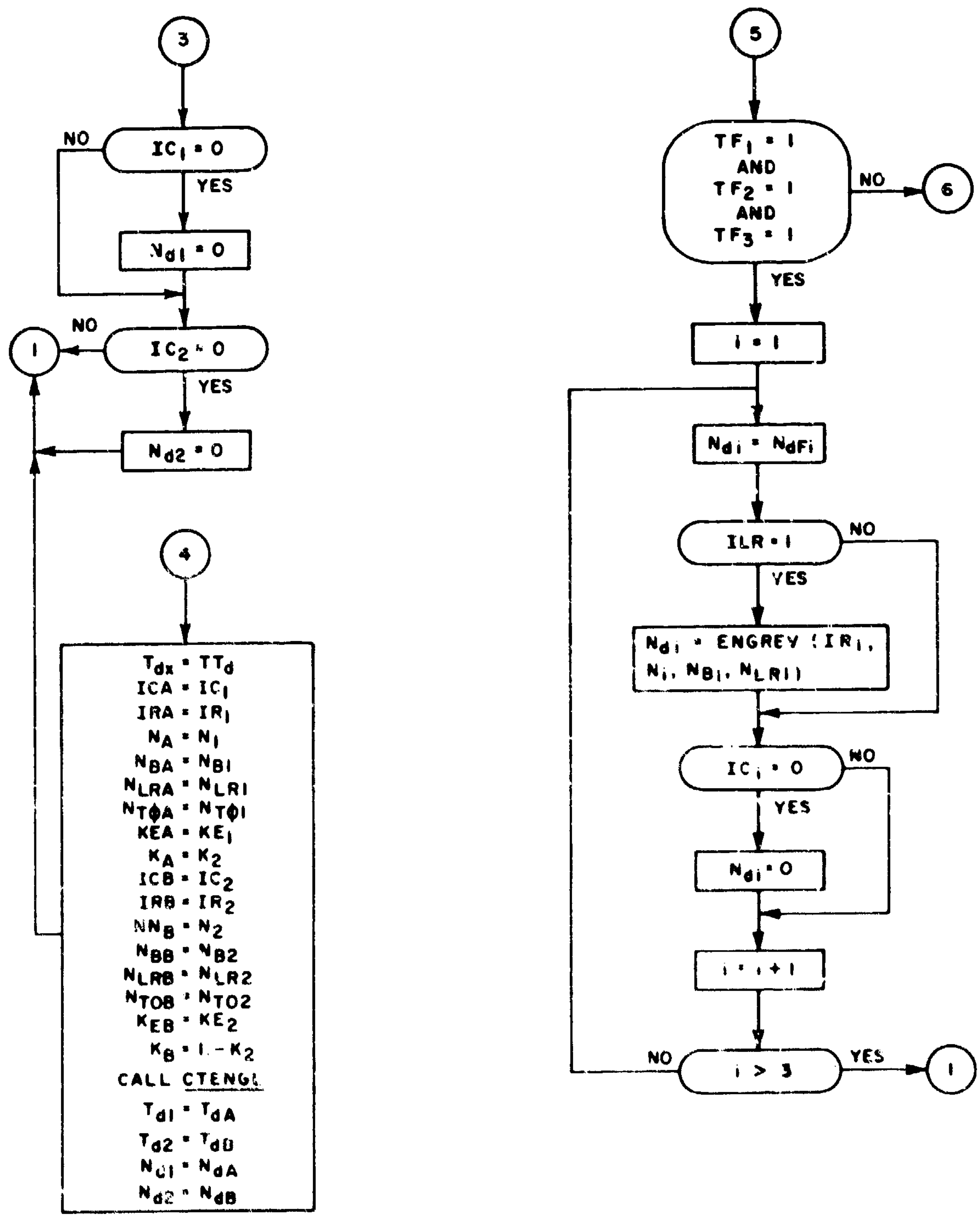
FLOW DIAGRAM (AUTPRI)



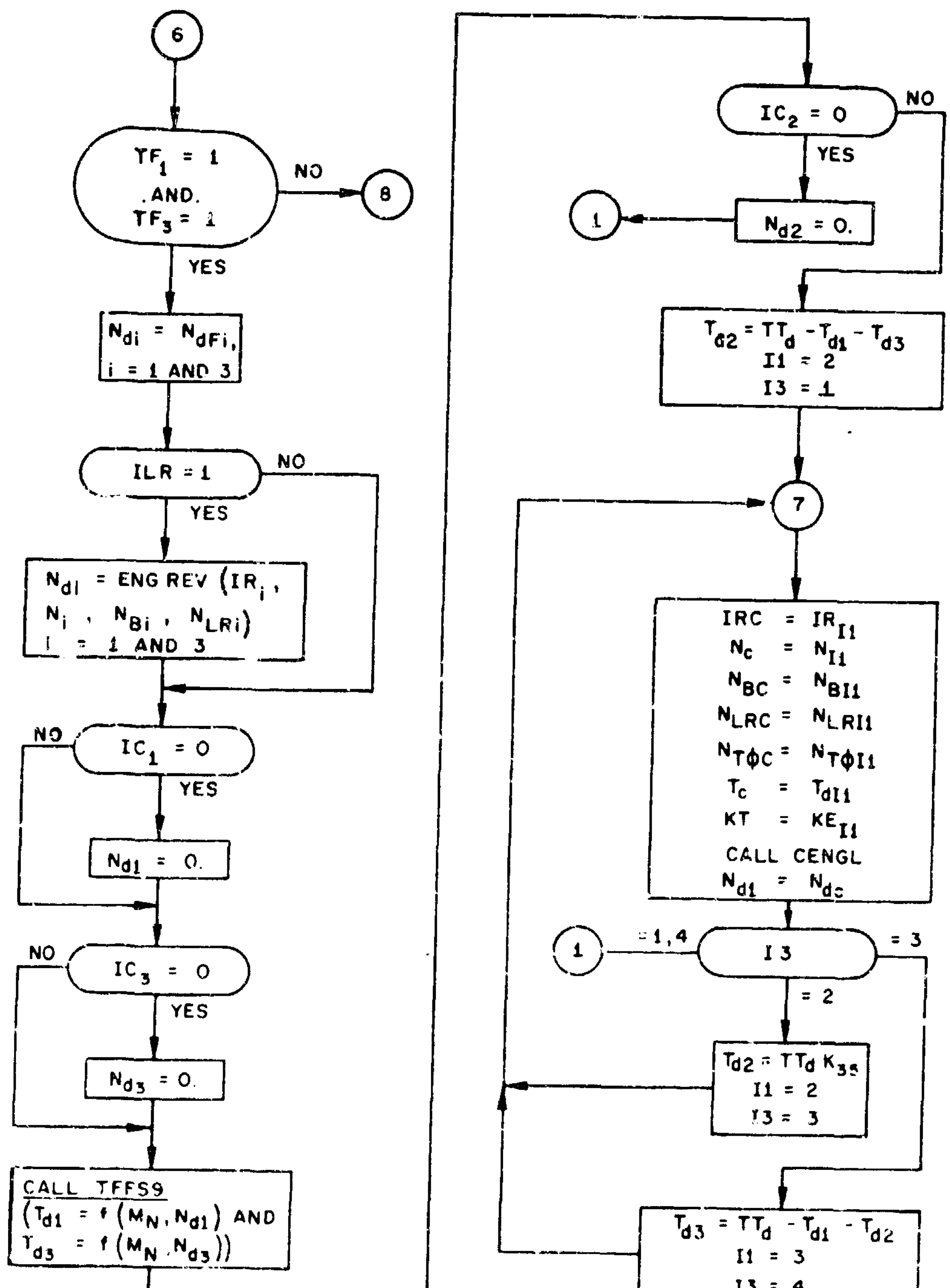
FLOW DIAGRAM (THAUTS)



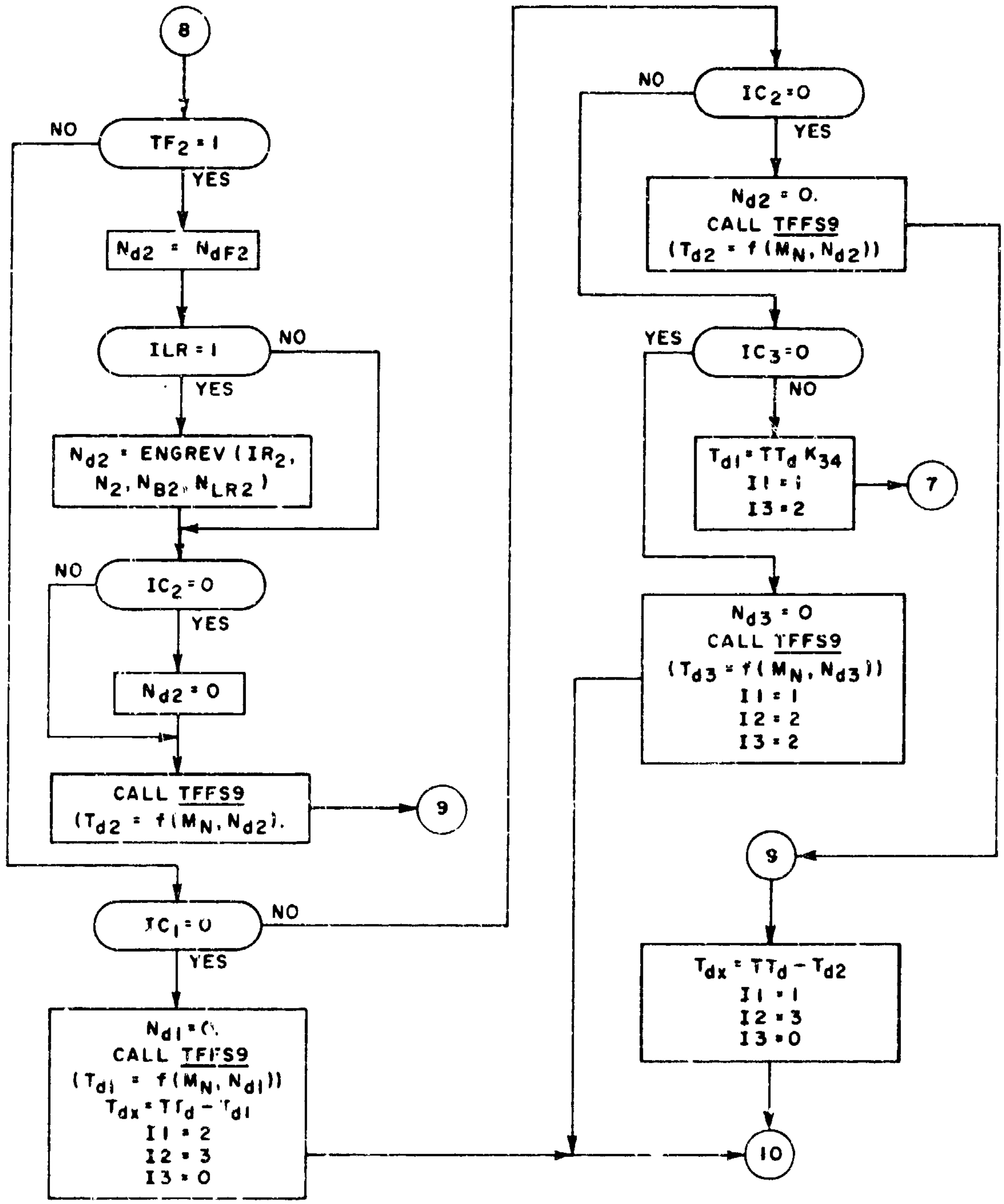
FLOW DIAGRAM (THAUTS)



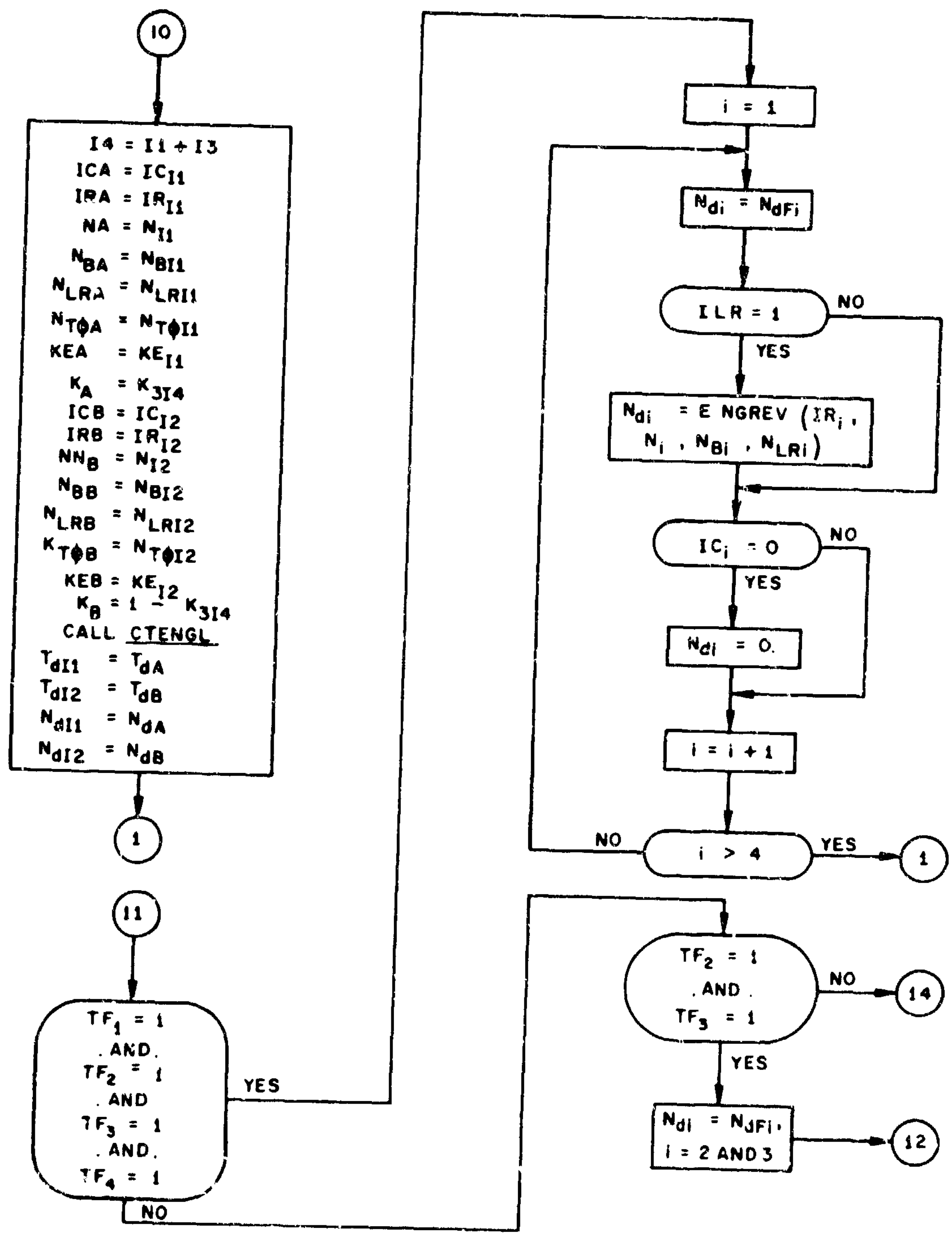
FLOW DIAGRAM (THAUTS)



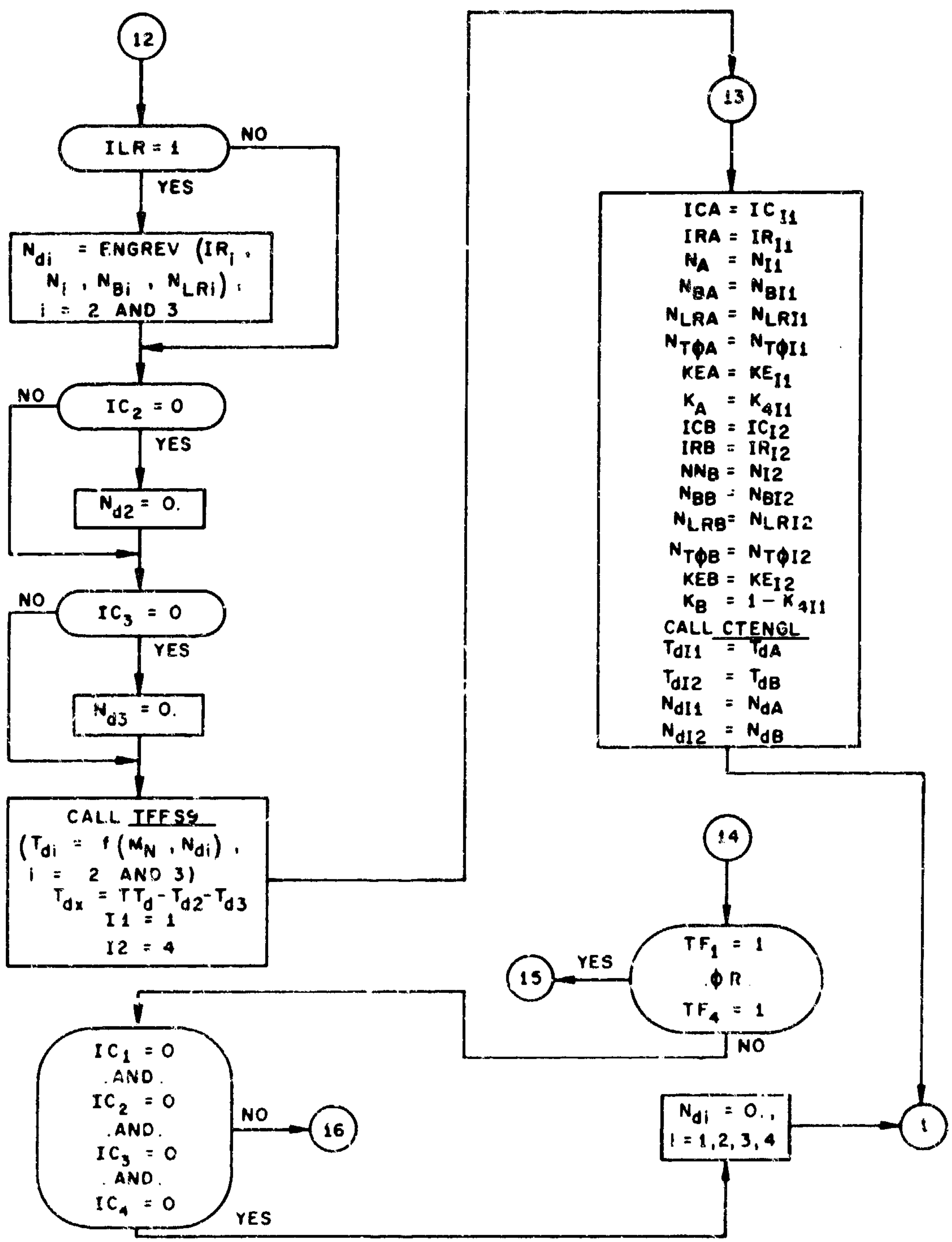
FLOW DIAGRAM (THAUTS)



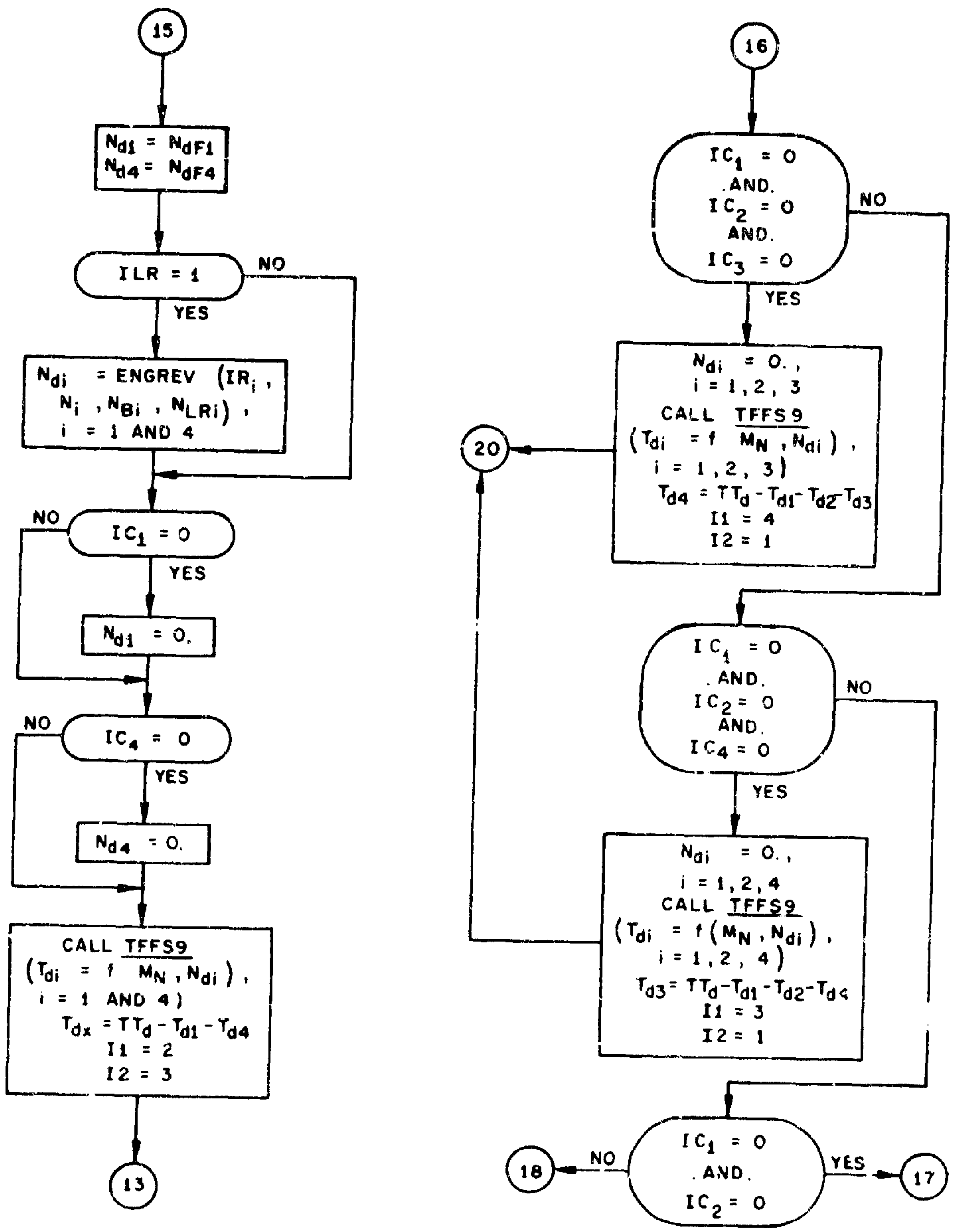
FLOW DIAGRAM (THAUTS)



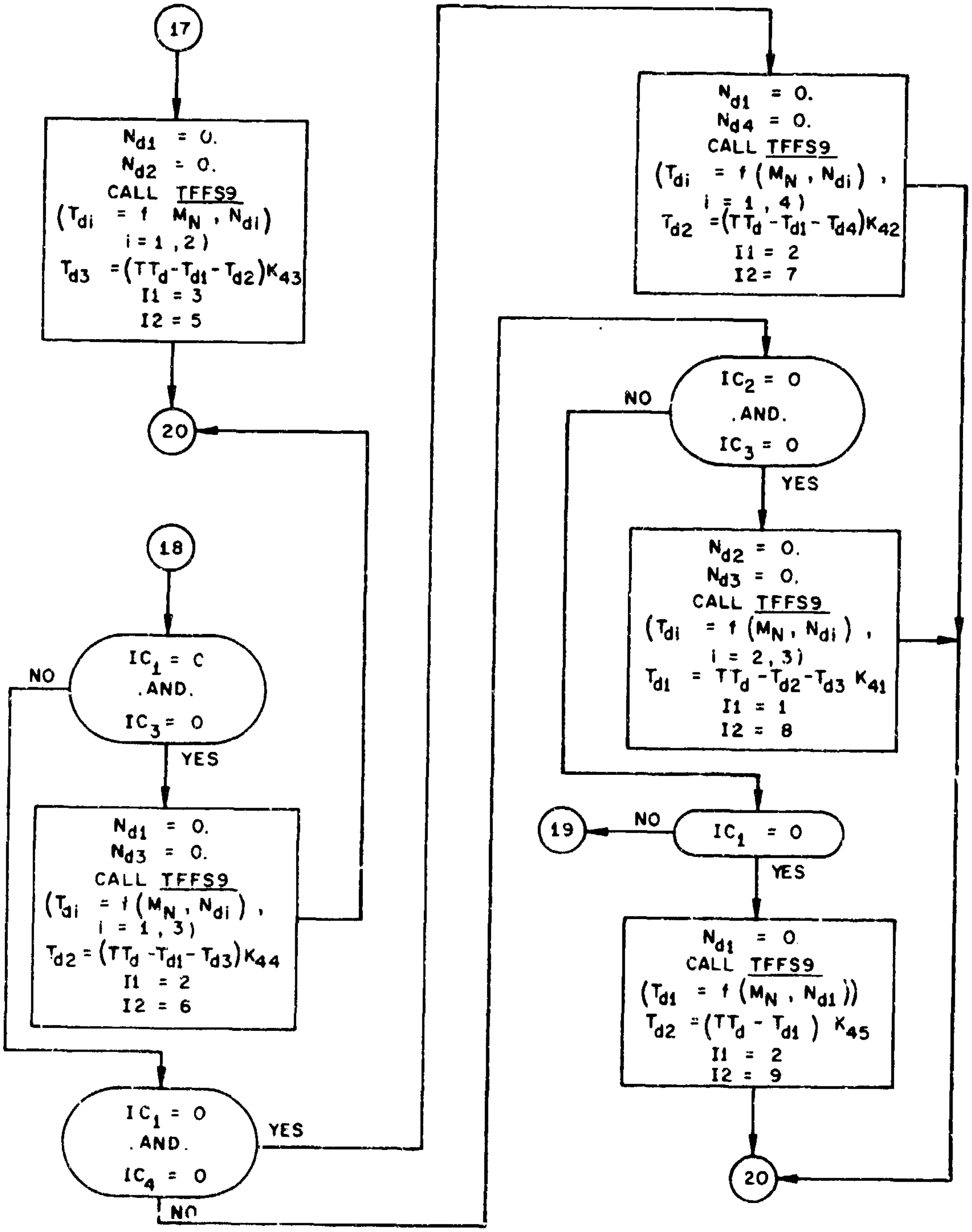
FLOW DIAGRAM (THAUTS)



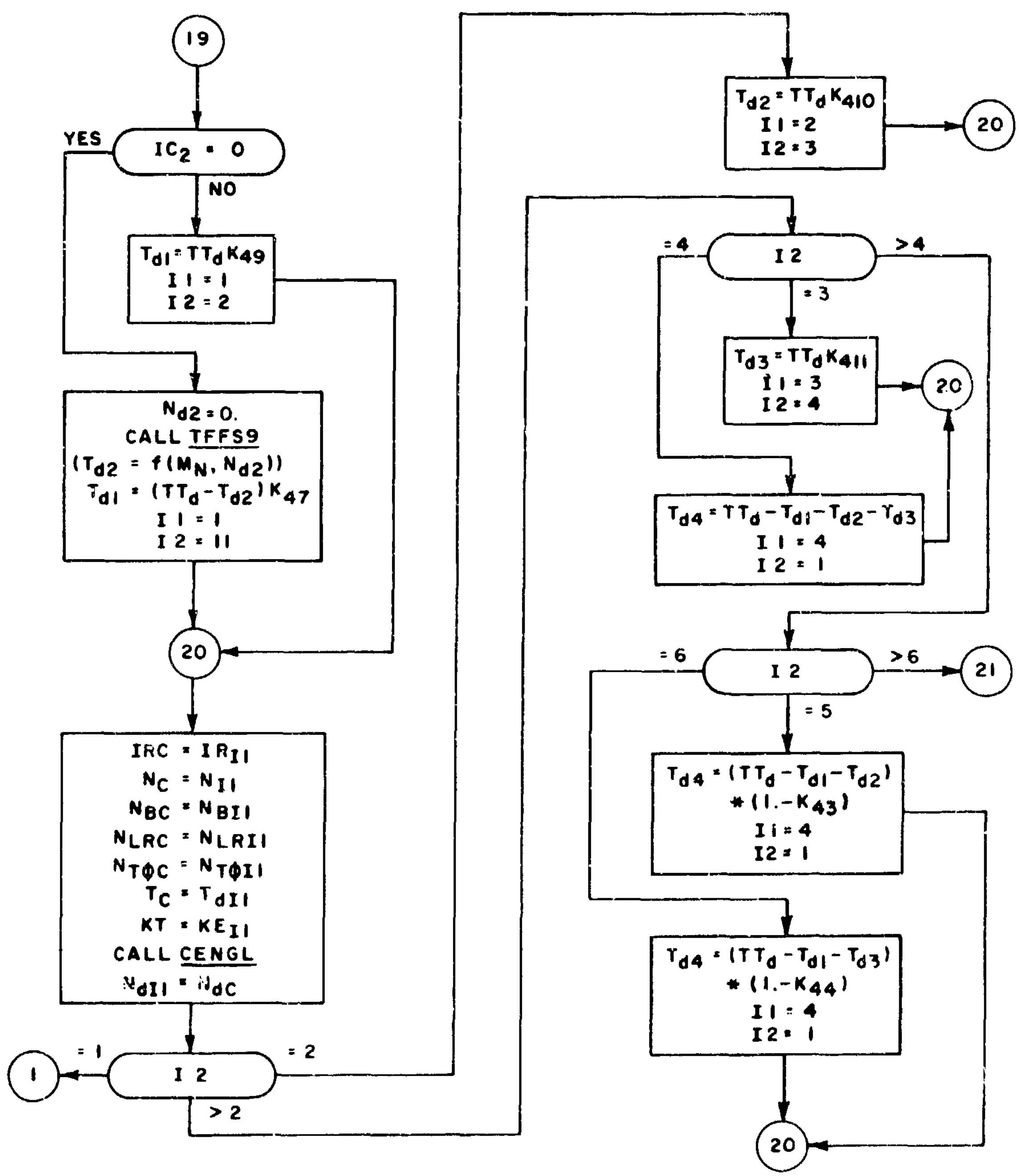
FLOW DIAGRAM (THAUTS)



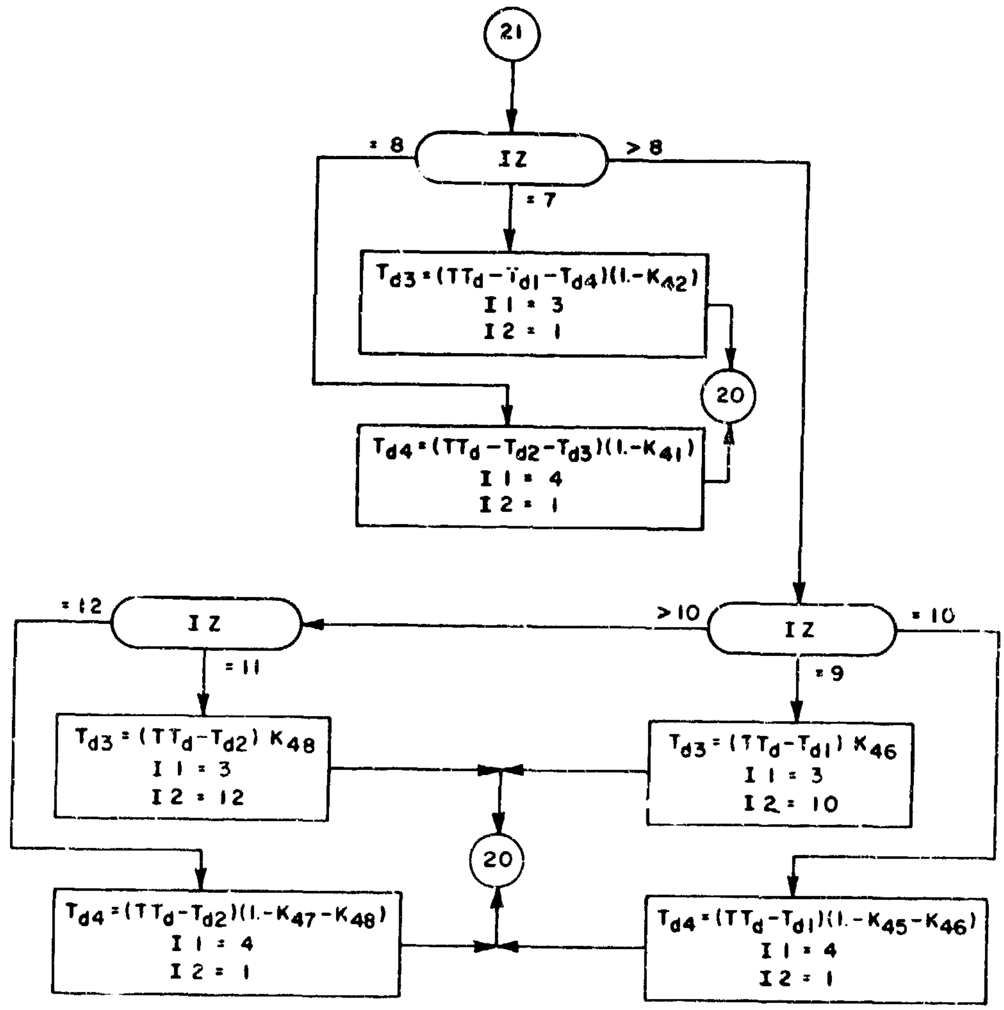
FLOW DIAGRAM (THAUTS)



FLOW DIAGRAM (THAUTS)



FLOW DIAGRAM (THAUTS)



6. ENGREV - AUTOPILOT ENGINE REVERSE LOGIC

a. Purpose - ENGREV is a function that determines if an engine is to be reversed.

b. Usage - Linkage to ENGREV is provided by the following statement:

$$Y = \text{ENGREV}\ (IR, N, N_B, N_{LR})$$

where

IR = engine reverse capability

N = engine throttle setting

N_B = engine throttle constraint

N_{LR} = reverse throttle setting for reverse signal on landing

7. ENGFL - AUTOPILOT ENGINE FAILURE LOGIC

a. Purpose - The ENGFL subroutine contains the logic to determine if an engine has failed or not.

b. Usage - Linkage to ENGFL is provided by the following statement:

```
CALL ENGFL
```

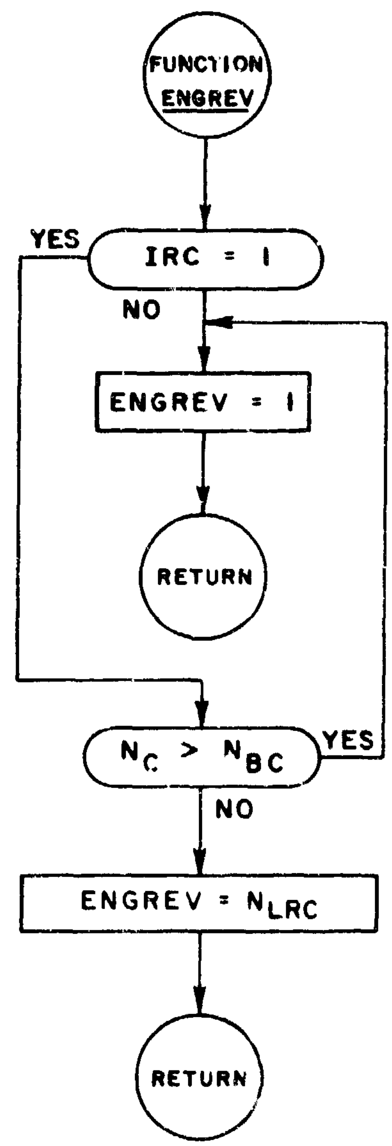
8. CTENGL - AUTOPILOT COMMON TWO-ENGINE LOGIC

a. Purpose - The CTENGL routine calculates the distribution of a total desired thrust load, $T_{\alpha x}$, between two engines, A and B, as T_{dA} and T_{dB}, respectively. It also determines the corresponding desired throttle settings, N_{dA} and N_{dB}, by calling CENGL twice.

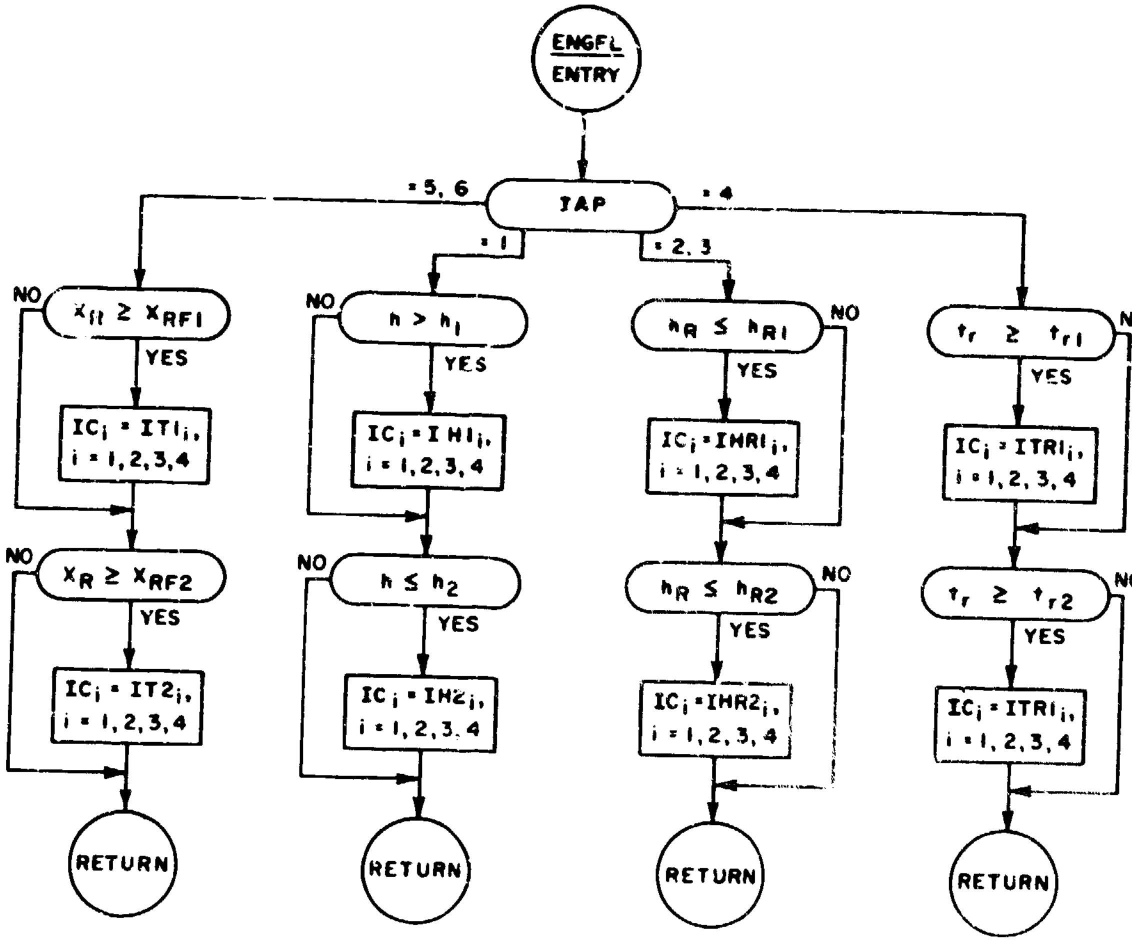
b. Usage - Linkage to CTENGL is provided by the following statement:

```
CALL CTENGL
```

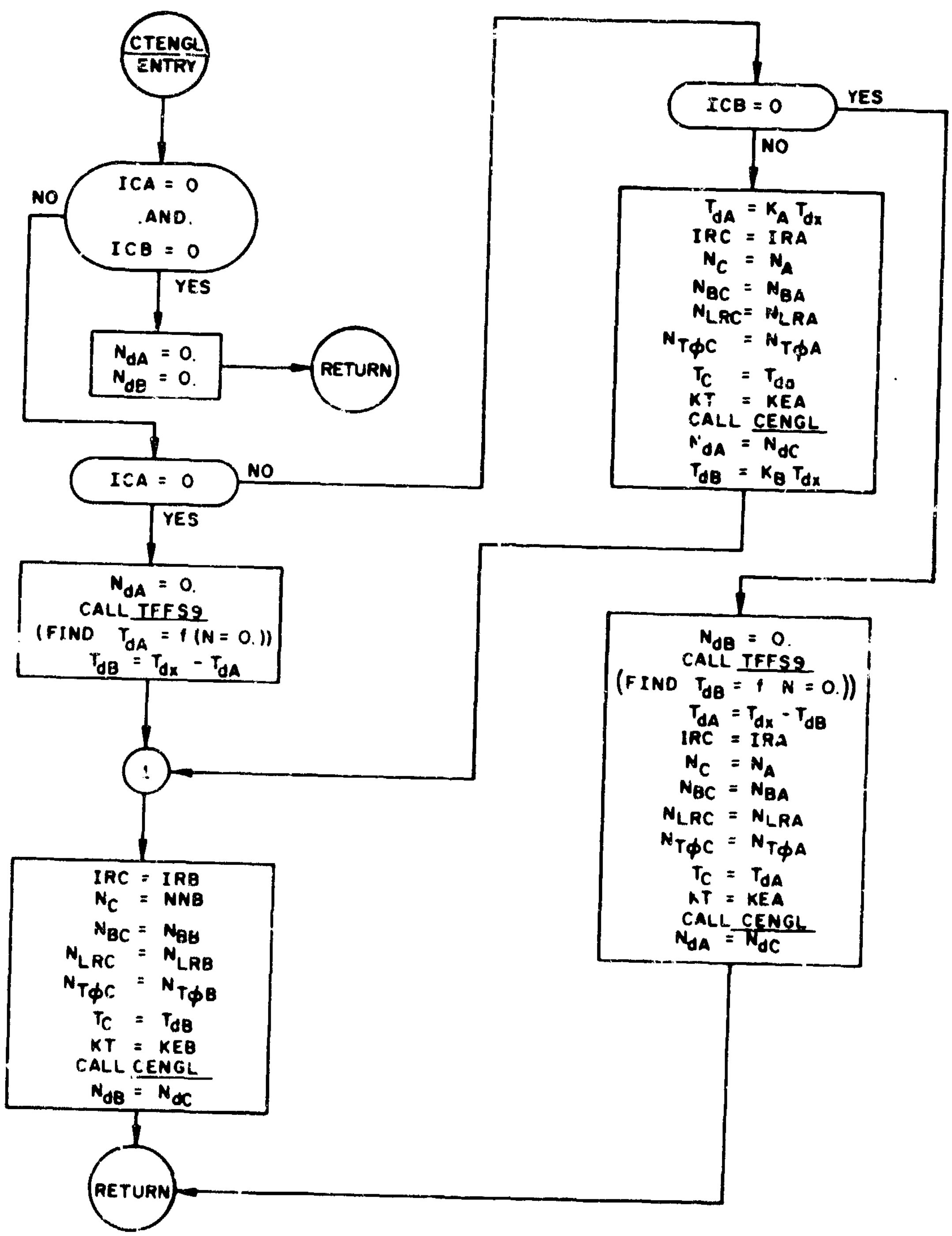
FLOW DIAGRAM (ENGREV)



FLOW DIAGRAM (ENGFL)



FLOW DIAGRAM (CTENGL)



9. CENGL - AUTOPILOT COMMON ENGINE LOGIC

a. Purpose - The CENGL routine calculates the variable N_{d_c} which is required by throttle autopilot (THAUTS)

b. Usage - Linkage to CENGL is provided by the following statement:

CALL CENGL

10. TFFS8 - THROTTLE SETTING SEARCH ROUTINE

a. Purpose - Search through ranges of throttle settings to find the throttle setting TN that corresponds with current thrust TC and Mach number.

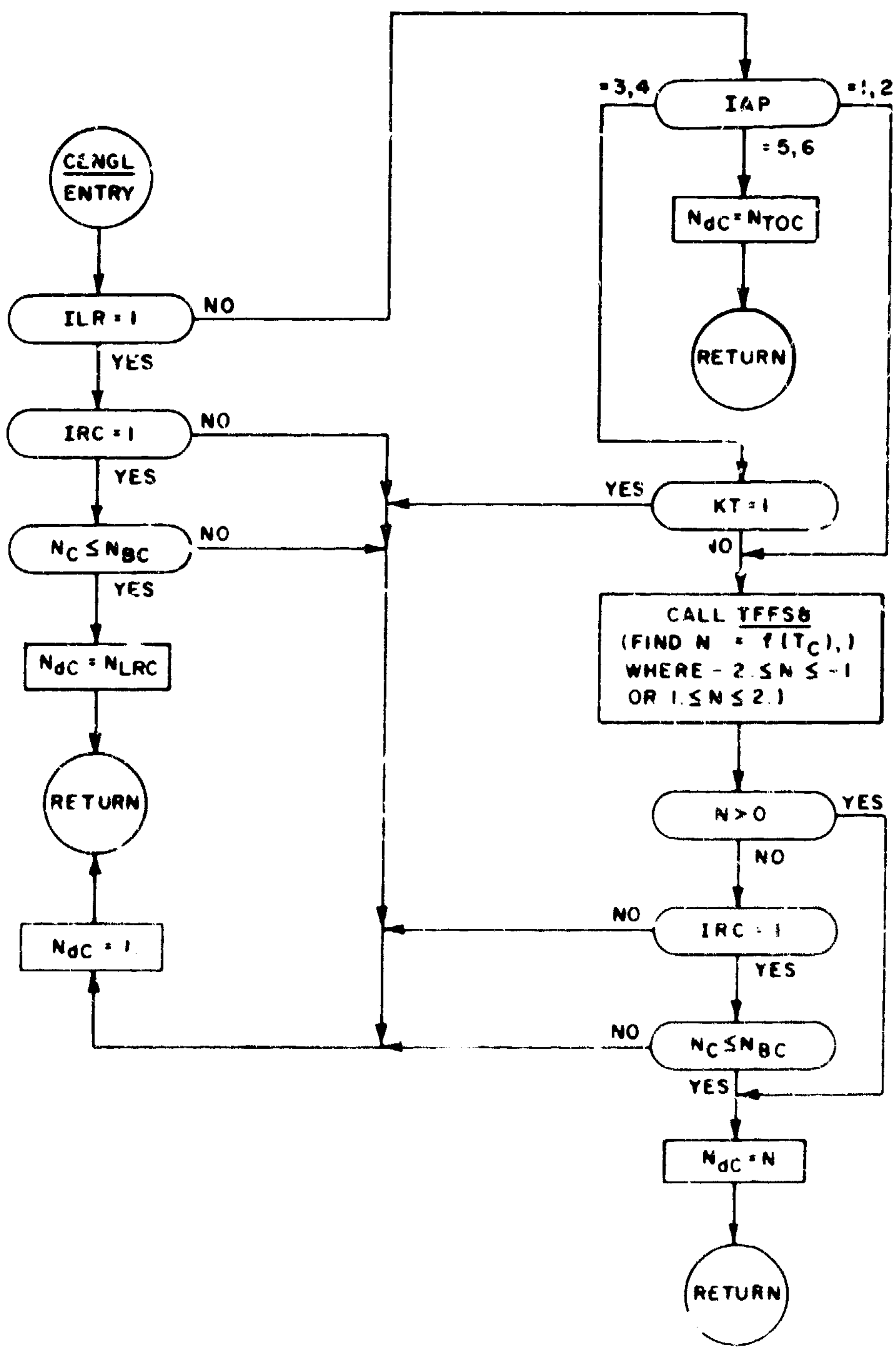
b. Usage - CALL TFFS8 (TC, TN)

11. TFFS9 - COMPUTE THRUST as F (MN, TN)

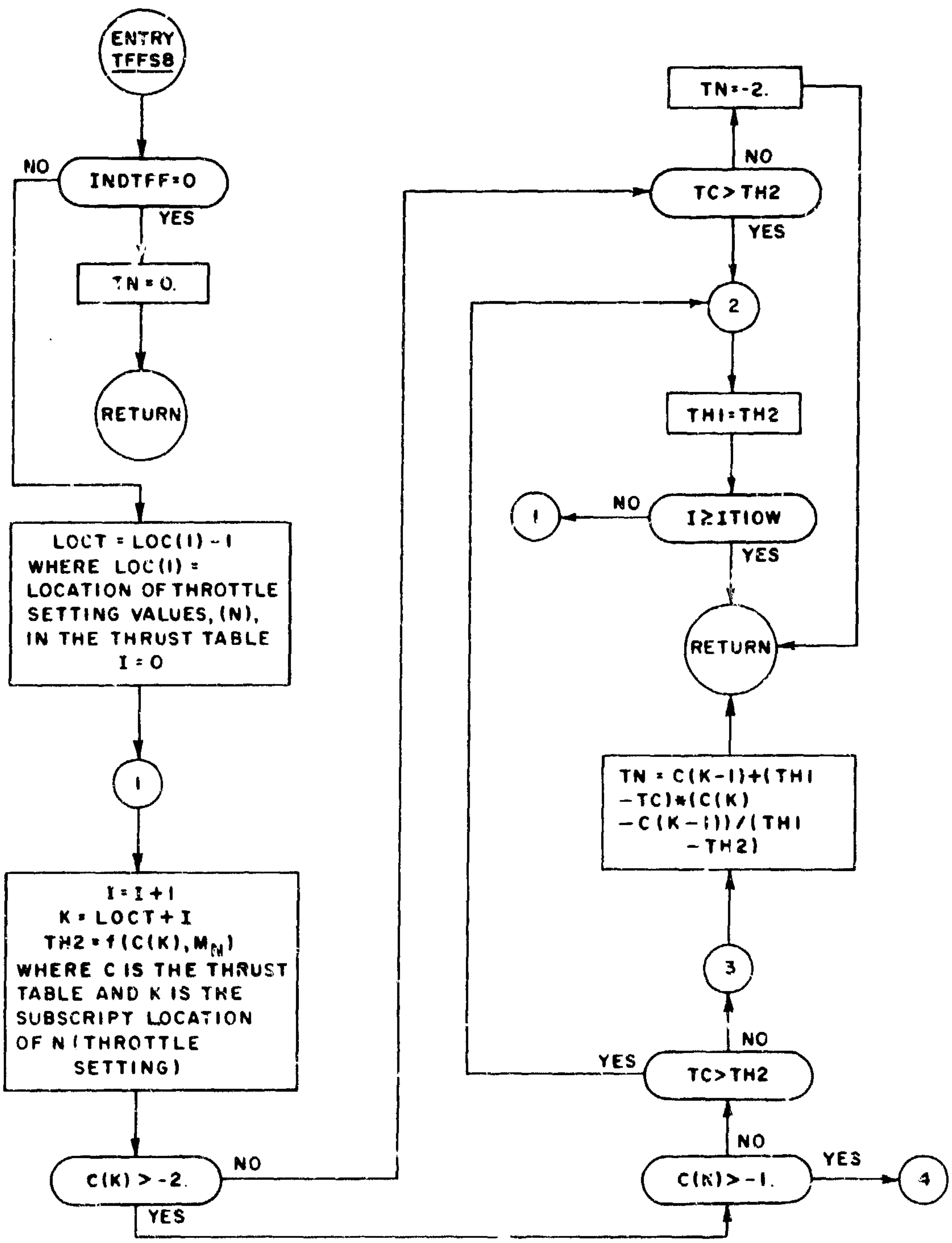
a. Purpose - Compute thrust as a function of current Mach number and designated throttle setting.

b. Usage - CALL TFFS9 (THRSET, THRUST)

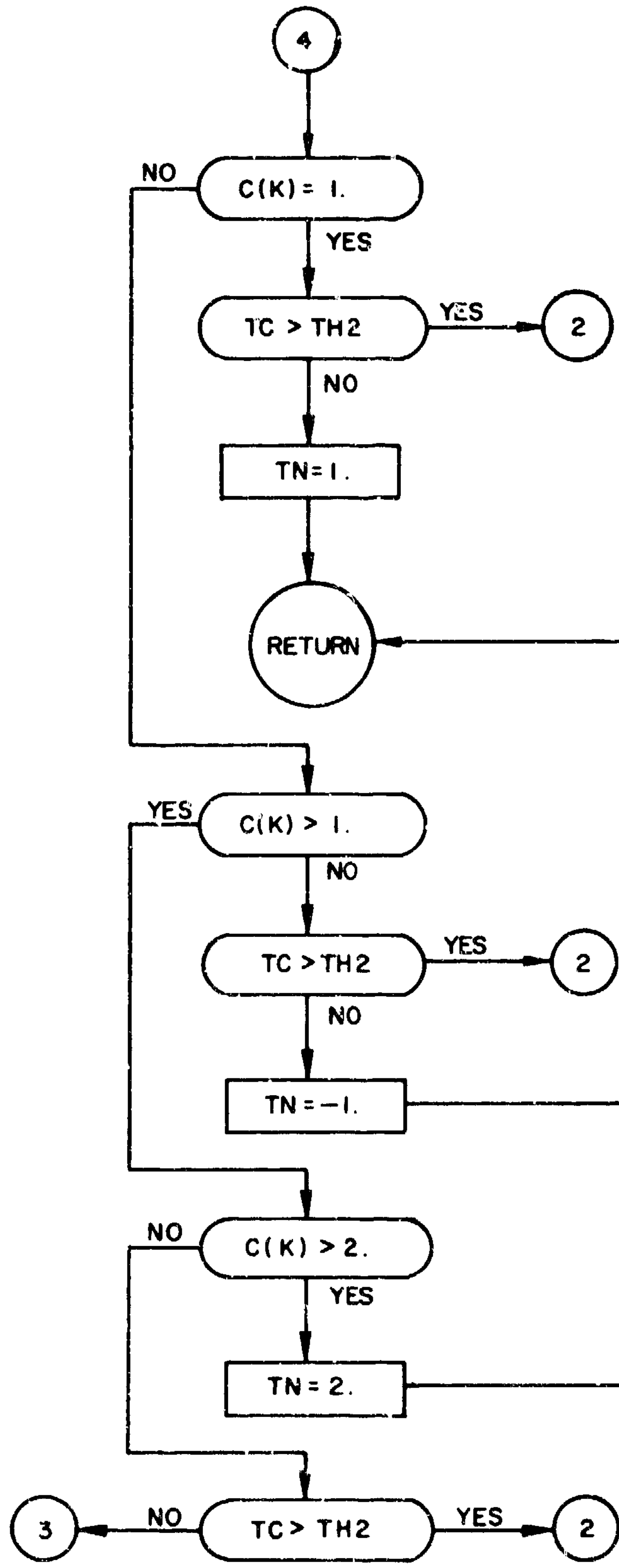
FLOW DIAGRAM (CENGL.)



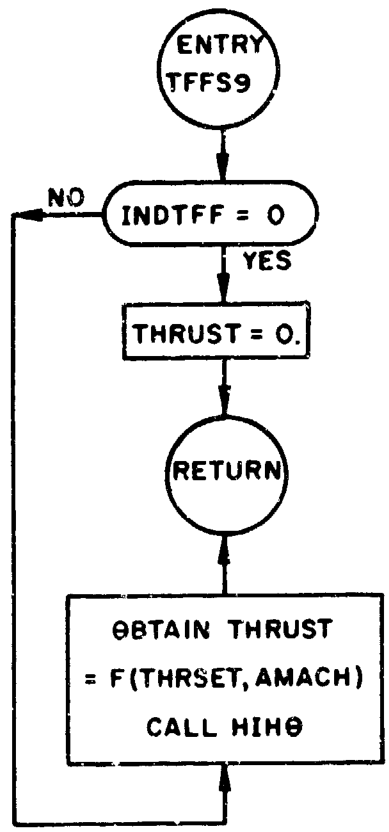
FLOW DIAGRAM (TFFSI)



FLOW DIAGRAM (TFFSI)



FLOW DIAGRAM (TFFS9)



SECTION VII

PLOT TAPE GENERATING PROGRAM (PLTSDF)

a. Purpose - To generate a plot tape from a data tape that was generated by the Takeoff and Landing Analysis Computer Program. The plot tape is plotted by the CALCOMP plotter.

b. Usage - After a plot tape has been generated by the TOLA program (see subroutine SDFLGP for data setup), this program is turned in as a s parate job.

c. Input: (1) TAPE - Data generated by TOLA program.

(2) CARDS - The following may be read in as input using the NAMELIST feature of FØRTRAN EXT. If the value of any variable is the same as its nominal value, it is not necessary to read it.

Name	Definition	Nominal Value
NCASES	= Number of sets of data or cases to be plotted	1
ISDFR	= 1 Rigid body data is stored on tape	1
	= 0 Rigid body data is not stored on tape	
ISDF	= 0 Do not plot rigid body data	0
	= 1 Plot rigid body data	
ISTPR(I)	= 1 Landing gear data for gear 1 stored on tape	1
	= 0 Landing gear data for gear 1 is not stored on tape	
ISTPL(I)	= 0 Do not plot landing gear data for landing gear 1	0
	= 1 Do not plot landing gear data for landing gear 1	
IL	= 1 Do not plot lower chamber pressure and 2nd Piston plots (2nd piston was not used)	0
	= 0 Do plot lower chamber pressure and 2nd piston plots	

Name	Definition	Nominal Value
TFIRST =	Trajectory time to begin plotting	0.
TLAST =	Trajectory time to stop plotting (Note= if both TFIRST = TLAST = 0., the entire time history on tape will be plotted)	0.
PLTINT = K	Plot every K^{th} point	1
FCTR =	The current factor all coordinates are multiplied by. (That is, the plot is made larger or smaller if FCTR is greater than 1. or less than 1. For example, if wish plot to be 25% of the original size, let FCTR=.25)	
XL =	Length of X - Axis in inches	7.2
YL =	Length of Y - Axis in inches	5.0

Examples of input data cards

Example No. 1: Plot rigid body variables and landing gear variables for gears 1, 3, 5. Plot every point and plot entire time history. Assume rigid body and landing gear variables for gears 1, 3, 5 are stored on tape.

```
EOR (7/8/9)
$INPUT ISDF=1,ISTPR=1,0,1,0,1,ISTPL=1,0,1,0,1$
EOJ (6/7/8/9)
```

Example No. 2: Plot landing gear variables for gear No. 3. Plot every other point from time = 4. to time = 10. seconds. Assume rigid body and landing gear variables for gears 1, 2, 3, 4, 5 are stored on tape for time = 0. to 20. seconds.

```
EOR (7/8/9)
$INPUT,ISTPL(3)=1,TFIRST=4.,TLAST=10.,PLTINT=2$
EOJ (6/7/8/9)
```

Example No. 3: Plot rigid body variables and landing gear variables for gear No. 5. Plot every point and plot entire time history. Assume rigid body and landing gear variables for gears 1, 3, 5 are stored on tape, with the size of graphs to be 50% of the original size where the original size of the X-Axis is 8 inches and the Y-Axis is 6 inches.

```
EOR (7/8/9)
$INPUT,ISDF=1,ISTPR=1,0,1,0,1,ISTPL(5)=1,XL=8.,YL=6.,FCTR=.50$
EOJ (6/7/8/9)
```

d. Output

TAPE. A plot tape is to be plotted by the CALCOMP plotter. The plot tape may be generated to give the following plots:

(a) Rigid body plots

Time (sec)	Vs	Ground Reaction roll moment (lb-ft)
"	Vs	Ground reaction pitch moment (lb-ft)
"	Vs	" " yaw moment (lb-ft)
"	Vs	Pitch rate (rad/sec)
"	Vs	Yaw angle (Deg)
"	Vs	Roll angle (Deg)
"	Vs	Z-Axis ground reaction (lb)
"	Vs	Z-Axis acceleration (ft/sec^2)
"	Vs	Mass Center sink rate (ft/sec)
"	Vs	Mass center altitude (ft)
Downrange (ft)	Vs	Cross range (ft)
"	Vs	Mass Center altitude (ft)
Time (sec)	Vs	Cross range (ft)
"	Vs	X-Axis acceleration (ft/sec^2)
"	Vs	X-Axis Velocity (ft/sec)

(b) Landing gear plots for each gear (1)

Time (sec) Vs ground reaction along strut for gear No. 1 (lb)

" Vs strut resistance for gear No. 1 (lb)

" Vs tire deflection for gear No. 1 (ft)

" Vs upper chamber pressure for gear No. 1 (lb/ft^2)

" Vs lower chamber pressure for gear No. 1 (lb/ft^2)

" Vs strut acceleration for gear No. 1 (ft/sec^2)

" Vs strut velocity for gear No. 1 (ft/sec)

" Vs strut displacement for gear No. 1 (ft)

" Vs 2nd piston acceleration for gear No. 1 (ft/sec^2)

" Vs 2nd piston velocity for gear No. 1 (ft/sec)

" Vs 2nd piston displacement for gear No. 1 (ft)

" Vs wheel axle moment for gear No. 1 (lb/ft)

" Vs tire angular acceleration for gear 1 (rad/sec^2)

" Vs tire angular rate for gear No. 1 (rad/sec)

(c) The following routines are called by SDFPLT:

PLØT	SCALE	PLØTE
PLØTS	LINE	READ
FACTOR	SYMBØL	BKSFIL
AXIS		

(d) The appropriate control cards are required after the job card to mount tape xxxx from TOLA on tape unit 13 and to mount tape yyyy for FLTSDF output (556 BPI) on tape unit 7.

(e) If TØLA and PLTSDF are run on the same job, it is not necessary to use tape unit 13 for storing data. In this case disk storage may be used.

☆U.S.Government Printing Office: 1975 — 657-020/510